Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO E IMPLEMENTACIÓN DE PRÁCTICAS PARA EL LABORATORIO DE COMUNICACIONES 4

Carlos Eduardo Pérez Álvarez
Asesorado por el Ing. Guillermo Antonio Puente Romero

Guatemala, febrero de 2015

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE PRÁCTICAS PARA EL LABORATORIO DE COMUNICACIONES 4

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

CARLOS EDUARDO PÉREZ ÁLVAREZ
ASESORADO POR EL ING. GUILLERMO ANTONIO PUENTE ROMERO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

GUATEMALA, FEBRERO DE 2015

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Narda Lucía Pacay Barrientos
VOCAL V	Br. Walter Rafael Véliz Muñoz
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADORA	Inga. María Magdalena Puente Romero
EXAMINADOR	Ing. Jorge Gilberto González Padilla
EXAMINADOR	Ing. Julio César Solares Peñate
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO E IMPLEMENTACIÓN DE PRÁCTICAS PARA EL LABORATORIO DE COMUNICACIONES 4

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 4 de febrero del 2013.

Carlos Perez

Carlos Eduardo Pérez Álvarez

Guatemala, 17 de septiembre de 2014.

Ing. Carlos Eduardo Guzmán Salazar
Coordinador de Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Ingeniero Guzmán:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado: **"DISEÑO E IMPLEMENTACIÓN DE PRÁCTICAS PARA EL LABORATORIO DE COMUNICACIONES 4"**, desarrollado por el estudiante Carlos Eduardo Pérez Álvarez con carné No. 2007-14411, ya que considero que cumple con los requisitos establecidos, por lo que el autor y mi persona somos responsables del contenido y conclusiones del mismo.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

Ing. Guillermo Antonio Puente Romero
ASESOR
Colegiado 5898

Guillermo A. Puente R.
INGENIERO ELECTRONICO
COL. # 5898

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Ref. EIME 48. 2014
Guatemala, 23 de OCTUBRE 2014.

Señor Director
Ing. Guillermo Antonio Puente Romero
Escuela de Ingeniería Mecànica Elèctrica
Facultad de Ingeniería, USAC.

Señor Director:

Me permito dar aprobación al trabajo de Graduación titulado: DISEÑO E IMPLEMENTACIÓN DE PRÁCTICAS PARA EL LABORATORIO DE COMUNICACIONES 4, del estudiante Carlos Eduardo Pérez Alvarez, que cumple con los requisitos establecidos para tal fin.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,
ID Y ENSEÑAD A TODOS

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

Ing. Carlos Eduardo Guzmán Salazar
Coordinador Área Electrónica

sfo

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

REF. EIME 48. 2014.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante; CARLOS EDUARDO PÉREZ ALVAREZ titulado: DISEÑO E IMPLEMENTACIÓN DE PRÁCTICAS PARA EL LABORATORIO DE COMUNICACIONES 4, procede a la autorización del mismo.

Ing. Guillermo Antonio Puente Romero

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 30 DE OCTUBRE 2,014.

Universidad de San Carlos
De Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.D.057-2015

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica al trabajo de graduación titulado: **DISEÑO E IMPLEMENTACIÓN DE PRÁCTICAS PARA EL LABORATORIO DE COMUNICACIONES 4,** presentado por el estudiante universitario: **Carlos Eduardo Pérez Álvarez** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
Decano

Guatemala, enero de 2015

/cc

ACTO QUE DEDICO A:

Dios	Por su infinito amor, por estar conmigo todos los días de mi vida y darme las fuerzas para alcanzar mis metas.
Mis padres	Miguel Pérez y Victoria Álvarez, por ser mi ejemplo a seguir, por su apoyo incondicional a lo largo de mi vida, lo cual permitieron alcanzar esta meta.
Mis hermanos	Cecilia y Miguel Pérez, por su apoyo y amor que siempre me han dado, que esto sea un ejemplo para que luchen por sus sueños.
Mis familiares	Abuelos, tíos, primos y sobrinos, por brindarme su apoyo en todo momento de mi vida.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por abrirme las puertas y permitirme ser parte de ella.
Ingeniero Guillermo Puente	Por su amistad y apoyo al realizar este trabajo de graduación, por sus consejos y confianza hacia mi persona.
Departamento de Matemática	Por brindarme un lugar de trabajo y experiencia, el cual me sirvió de apoyo para poder realizar este trabajo de graduación y a todas las personas que allí laboran, por brindarme su amistad.
Mis amigos	Por estar en las buenas y malas, por su apoyo incondicional y cariño para la realización de este trabajo.
Mis compañeros	Por todos los momentos y proyectos compartidos, desvelos y demás a lo largo de esta carrera.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
RS-232	Estándar recomendado 232
Fs	Frecuencia de muestreo
Hz	Hertz
KHz	Kilo Hertz
Ts	Período de muestreo
To	Período fundamental
X(t)	Señal continua
X(k)	Señal discreta
FFT	Transformada rápida de Fourier
Rms	Valor cuadrático medio

GLOSARIO

ADC Analog to Digital Conversion (Conversión analógica a digital). Consiste en la transcripción de señales analógicas en señales digitales.

Algoritmo Conjunto de pasos o instrucciones ordenadas y simples que permiten definir de manera más fácil un procedimiento a ejecutar.

Amplificador Dispositivo electrónico que mediante el uso de energía, aumenta la magnitud de una señal.

Armónicos Son frecuencias múltiples mayores a la frecuencia fundamental, cuya amplitud va decreciendo conforme el múltiple.

ASCII American Standard Code for Information Interchange (Código Estándar Estadounidense para el Intercambio de Información).

Basic Familia de lenguajes de programación de alto nivel.

Bit Unidad básica de información equivalente entre dos estados lógicos igualmente probables.

Byte Conjunto de información formado por 8 bits.

Circuito integrado	Pastillas pequeñas de material semiconductor, sobre la que se fabrican gran cantidad de circuitos electrónicos.
Codificación	Proceso de conversión de un sistema de datos de origen a otro sistema de datos de destino.
Comunicación serial	Comunicación en la cual los datos de información viajan a través de bits y solamente un bit a la vez en el mismo medio.
Convolucion	Es un operador matemático que transforma dos funciones *f* y *g* en una tercera función que en cierto sentido representa la magnitud en la que se superponen *f* y una versión trasladada e invertida de *g*.
Copilador	Programa que convierte el lenguaje informático de alto nivel empleado por el usuario en lenguaje binario.
Cuantizacion	Es la conversión de una señal discreta en el tiempo, evaluada de forma continúa a una señal discreta en el tiempo discretamente evaluada.
DAC	Digital to Analogue Converter (Conversor Digital analógico). Un dispositivo para convertir señales digitales con datos binarios en señales de corriente.

DC	Corriente directa.
Dip	Encapsulamiento en línea dual.
DSP	Digital Signal Processor (Procesador Digital de Señales). Es un sistema basado en un microprocesador que posee un conjunto de instrucciones.
Estimación espectral	Es una función matemática que informa de cómo está distribuida la potencia o la energía en las distintas frecuencias de las que está formada.
FIR	Finite Impulse Response (Respuesta Finita al Impulso). Es un filtro digital que como respuesta a un impulso tendrá a la salida un número finito de términos no nulos.
Filtro	Es un dispositivo que discrimina una determinada frecuencia de una señal eléctrica que pasa a través de él, con el cual puede cambiar magnitud y fase.
GUI	Graphic User Interface (Interfaz Gráfica de Usuario) Conjunto de formas y métodos que posibilitan la interacción de un sistema con los usuarios utilizando formas gráficas e imágenes.
Hardware	Conjunto de componentes físicos de un sistema.

IIR	Infinite Impulse Response (Respuesta Infinita al Impulso). Es un filtro digital que como respuesta a un impulso tendrá a la salida un número infinito de términos no nulos.
Impulso	Es la variación, generalmente breve (unos pocos microsegundos) en intensidad o tensión de una corriente pulsante.
Interfaz	Conexión entre dos computadoras o máquinas de cualquier tipo dando una comunicación entre ambas.
Microchip	Circuito integrado en miniatura, que realiza numerosas funciones en dispositivos electrónicos.
Microcontrolador	Circuito integrado de gran escala de integración, que incorpora la mayor parte de los elementos que configuran un controlador.
Mips	Millones de instrucciones por segundo.
Multiplexacion	Es la combinación de 2 o más canales de información en un solo medio de trasmisión, usando un dispositivo llamado multiplexor.
Período	Intervalo de tiempo mínimo entre dos posiciones idénticas de una partícula con movimiento oscilatorio.

PIC	Peripheral Interface Controller (Controlador de interfaz periférico). De la familia de microcontroladores tipo RISC fabricados por Microchip.
PWM	Pulse Width Modulation (Modulación por Ancho de Pulsos). Es una técnica donde se modifica el ciclo de trabajo de una señal periódica.
Respuesta al impulso	Es la respuesta de un sistema cuando la entrada es un impulso y como salida se tiene una descripción equivalente a la dada por una función de trasferencia.
Sensor	Dispositivo capaz de trasformar magnitudes físicas o químicas en variables eléctricas.
Software	Conjunto de componentes lógicos de un sistema.
Trasformada de Fourier	Espectro de frecuencias de una función.
Trasformada Z	Convierte una señal real o compleja definida en el dominio del tiempo discreto en una representación en el dominio de la frecuencia compleja.

UART	Universal Asynchronous Receiver-Trasmitter (Transmisor-Receptor Asincrono Universal) Realiza el control sobre los dispositivos que se manejan en el puerto serial.
Variable continua	Es aquella que toma valores en uno o varios intervalos de la recta real.
Variable discreta	Es aquella que sólo puede tomar valores dentro de un conjunto finito, como los números naturales.

RESUMEN

El presente trabajo de investigación describe una serie de prácticas de laboratorio utilizando MATLAB y el dsPIC30F4013, para el Laboratorio de Comunicaciones 4 de la carrera de Ingeniería en Electrónica.

Las prácticas aquí propuestas se basan en el tema del procesamiento digital de señales, que es lo que se estudia de una forma teórica en el curso de Comunicaciones 4. La finalidad del siguiente trabajo es poder mostrar al estudiante la parte práctica de dicho curso y así podrá desarrollar habilidades tanto teóricas como prácticas en dicho tema.

En el primer capítulo se describen las características generales y herramientas matemáticas que son útiles para el procesamiento digital de señales. El segundo describe el software utilizado para el desarrollo de prácticas de laboratorio propuestas en el capítulo 4. El tercer capítulo describe el microcontrolador de procesamiento digital de señales dsPIC, sus características y aplicaciones, así como la placa entrenadora utilizada para las prácticas de laboratorio.

El cuarto capítulo describe 3 prácticas utilizando MATLAB. En el quinto capítulo presenta 5 prácticas utilizando el dsPIC30F4013, donde se realizan dichas prácticas paso a paso para mostrar de una forma didáctica la utilización de este microcontrolador y sus aplicaciones en el procesamiento digital de señales. En el último capítulo se proponen 3 prácticas más con diferentes aplicaciones, donde se puede utilizar también un dsPIC.

.

OBJETIVOS

General

Diseñar e implementar prácticas para el Laboratorio de Comunicaciones 4, utilizando MATLAB y el dsPIC30F4013.

Específicos

1. Presentar una introducción al procesamiento digital de señales, así como sus características y las herramientas matemáticas que son útiles para la manipulación e interacción con dicho tema.

2. Presentar el software aplicado al procesamiento digital de señales como lo es MATLAB, sus características y aplicaciones.

3. Realizar un estudio y análisis del microcontrolador de procesamiento digital de señales dsPIC, con el cual se puede realizar muchas aplicaciones para el procesamiento digital de señales.

4. Presentar una propuesta de desarrollo de prácticas de laboratorio, utilizando el software MATLAB.

5. Presentar una propuesta de desarrollo de prácticas de laboratorio, utilizando el microcontrolador dsPIC30F4013.

6. Mostrar otras aplicaciones utilizando dsPIC30F4013.

INTRODUCCIÓN

En la actualidad la tecnología avanza a pasos agigantados, donde se vive en un mundo de continuo cambio, por ello el ingeniero tiene que ir al ritmo de esta evolución. A esto se debe la importancia de que la formación en el tema del procesamiento digital de señales, sea lo más sólido posible.

En la actualidad el curso de Comunicaciones 4, perteneciente a la carrera de Ingeniería en Electrónica de la Facultad de Ingeniería, es donde el estudiante adquiere conocimiento teórico sobre el procesamiento digital de señales; como lo son los temas, trasformada de Fourier, trasformada z, el diseño de filtros digitales FIR e IIR entre otros tópicos importantes.

El siguiente trabajo tiene la misión de proponer una serie de prácticas de laboratorio utilizando software como hardware, para el laboratorio del curso Comunicaciones 4. En dicho trabajo se plantea de una forma didáctica y ordenada los pasos para realizar cada práctica, así como los algoritmos y circuitos utilizados.

El dsPIC30F4013 y la placa entrenadora EasydsPIC4A que es el hardware con el que se realizó las prácticas de laboratorio del presente trabajo son proporcionados por el Laboratorio de Electrónica de la Facultad de Ingeniería, aquí se puede encontrar también computadoras con el software necesario para la programación de dicho microcontrolador.

1. INTRODUCCIÓN AL PROCESAMIENTO DIGITAL DE SEÑALES

El estudio del procesamiento de señales digitales, se ha ido profundizando cada día más, desde su aparición en la década del 70. En la actualidad, es de suma importancia conocer cada detalle de este interesante campo.

En este primer capítulo se estudian las bases del procesamiento digital de señales, sus propiedades, sus diferentes representaciones, así como su representación matemática y sus aplicaciones.

1.1. Procesamiento digital de señales

En el principio el mayor interés en este campo era el desarrollo de algoritmos para la transformada de Fourier y el diseño de filtros digitales.

En la actualidad un ingeniero en el área de telecomunicaciones debe poseer una buena base de conocimientos en las siguientes áreas: procesos estocásticos, teoría de matrices, algoritmos avanzados y sistemas dinámicos entre otras.

Muchos pensaran que estos tópicos parecieran ser cosa de investigadores. Es una realidad que un ingeniero actual en el área se enfrenta a retos como: diseñar sistemas de filtraje óptimo, filtraje adaptivo y estimación espectral. Con esto se concluye que los tópicos mencionados antes son parte fundamental en la formación básica de un ingeniero en la actualidad.

En la actualidad se considera que un curso introductorio en el procesamiento digital de señales, debe incluir los siguientes tópicos: transformada Z, respuesta al impulso, convolución, respuesta a la frecuencia, el teorema de muestreo, transformada discreta de Fourier, algoritmos de trasformada rápida de Fourier, diseño de filtros de respuesta finita al impulso (FIR) y diseño de filtros de respuesta infinita al impulso (IIR). Todos estos son tópicos vistos en el curso de Comunicaciones 4.

Es importante mencionar que dado que estos temas son bien conocidos, existe ya un buen número de paquetes de software que manejan este material estándar, por ejemplo: MATLAB (el cual será utilizado en este trabajo) por mencionar alguno y que pueden servir como un soporte paralelo en el estudio de dichos temas, sobre todo en cuanto a aplicaciones.

1.2. Señales y sistemas

Los conceptos de señales y sistemas se pueden observar en una diversidad de campos, de manera que las ideas y técnicas asociadas con estos conceptos son importantes en las siguientes áreas; comunicaciones, aeronáutica, diseño de circuitos, acústica, óptica, sismología, ingeniería biomédica, sistemas de generación y distribución de energía, control de procesos, biométrica, entre otras.

Si bien la naturaleza física de las señales y sistemas que aparecen en estas áreas pueden ser diferentes, todas estas tienen en común dos características básicas: las señales y los sistemas.

Las señales son funciones de una o más variables independientes y contienen información acerca de la naturaleza o comportamiento de algún fenómeno, los sistemas reciben señales como entrada y responden a ellas produciendo otras señales a la salida.

Se puede observar la relación entre señales y sistemas en el diagrama de bloques de la figura 1.

Figura 1. **Diagrama de bloques de un sistema general**

Fuente: elaboración propia, con programa Microsoft Word 2010.

El voltaje y la corriente como funciones del tiempo, aplicados a un circuito son ejemplo de señales y el circuito es un sistema, el cual responderá con una señal de corriente o voltaje según sea lo aplicado.

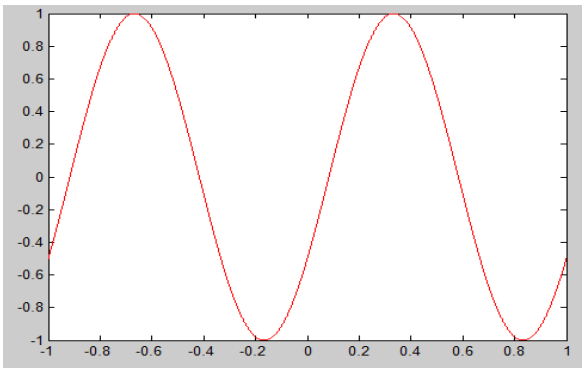
A continuación se introduce una descripción y representación matemática de señales y sistemas que permitirá involucrar los conceptos básicos, para así posteriormente poder utilizarlos en las prácticas que se presentaran en capítulos más adelante, que son el tema principal de este trabajo.

1.2.1. Señales continuas y discretas

Aunque las señales se pueden representar matemáticamente como funciones de una o más variables independientes, aquí se tratará exclusivamente el caso de funciones de una variable independiente y esta variable normalmente será el tiempo.

Existen 2 tipos básicos de señales: de tiempo continuo y de tiempo discreto. En una señal continua o señal de tiempo continuo $x(t)$, la variable independiente (tiempo) es una variable continua, la cual se denotará en este trabajo por la letra t, a la variable independiente. La figura 2 es un ejemplo de señales de tiempo continuo.

Figura 2. **Señal en tiempo continuo**

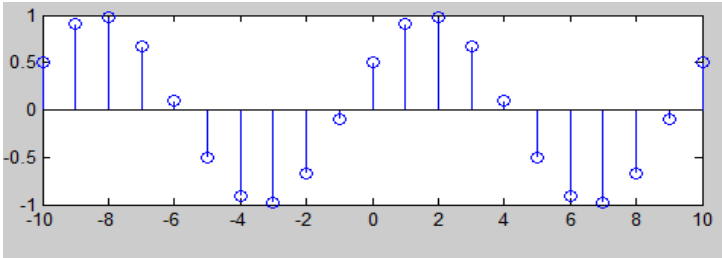


Fuente: elaboración propia, con programa MATLAB R2010b.

Por otro lado, una señal discreta o señal de tiempo discreto $x(k)$, solamente está definida en ciertos instantes discretos de tiempo, de manera que entre cada instante y el siguiente no está definida dicha señal.

Una señal de tiempo discreto también se puede por lo tanto representar como una lista o secuencia de valores $\{x(1), x(2), x(3) \ldots\}$. En este tipo de señales se usará k para denotar la variable independiente. La figura 3 es un ejemplo de señal de tiempo discreto.

Figura 3. **Señal tiempo discreto**



Fuente: elaboración propia, con programa MATLAB R2010b.

1.2.2. Sistemas

Un sistema se puede ver como cualquier proceso que produce una transformación de señales. Todo sistema debe tener al menos una entrada *x* y una salida *y*, la señal de la salida está relacionada con la entrada mediante una relación de transformación $y = f(x)$.

De manera similar a como se hizo con las señales, los sistemas pueden ser de tiempo continuo si transforma señales de entrada de tiempo continuo, en señales de salida de tiempo continuo y serán llamados sistemas de tiempo discreto si transforman señales de entrada de tiempo discreto en señales de salida de tiempo discreto.

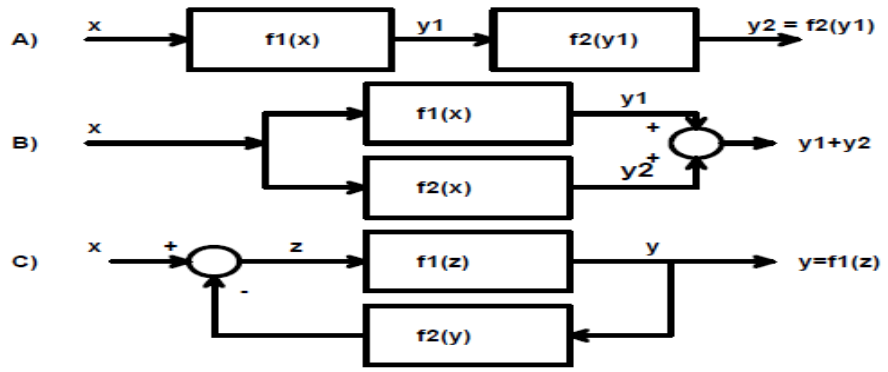
1.2.2.1. Interconexión de sistemas

Los diagramas de bloques permiten representar operaciones básicas entre sistemas, esto es su interconexión, la cual puede ser de tres tipos:

- Interconexión en serie o cascada
- Interconexión en paralelo
- Interconexión en retroalimentación

Estos tipos de interconexión son mostrados en la figura 4.

Figura 4. **Interconexión de sistemas a) cascada b) paralelo c) retroalimentación**



Fuente: procesamiento de señales digitales. http://www.angelfire.com/falcon/shadowrsv/tsenales/curdsp1. Consulta: 10 de abril de 2013.

1.2.2.2. Los sistemas y sus propiedades

A continuación se describen algunas de las propiedades más importantes de los sistemas. Estas tienen interpretaciones tanto físicas como matemáticas.

Son propiedades muy generales, es decir no atienden a la naturaleza física del sistema en sí, el cual puede ser eléctrico, químico o mecánico. Sino más bien al tipo de efecto que se aplica a las señales.

1.2.2.3. Sistemas con y sin memoria

Un sistema se dice sin memoria si su salida en un instante dado, depende de su entrada solamente en ese instante (un sistema de este tipo en ocasiones es llamado sistema estático).

Un sistema cuya salida puede depender de entradas en instantes anteriores al actual se denomina sistema con memoria. Este tipo de sistemas también suele llamarse sistema dinámico.

1.2.2.4. Causalidad

Un sistema es causal si su salida en cualquier instante depende solo de los valores de la entrada en el instante actual o en instantes anteriores. A este tipo de sistemas también se le llama no anticipativo, ya que la salida del sistema no anticipa valores futuros de la entrada.

Una consecuencia fundamental de que un sistema sea causal, es el hecho de que si dos entradas a un sistema causal son idénticas desde las condiciones iniciales hasta un instante t_o, las salidas correspondientes también serán iguales hasta ese mismo instante.

1.2.2.5. Estabilidad

Intuitivamente un sistema estable es aquel en que entradas pequeñas producen salidas que no divergen, es decir; salidas acotadas.

1.2.2.6. Invariante en el tiempo

Un sistema se dice invariante en el tiempo si un retardo en la señal de entrada produce una señal de salida retardada en la misma cantidad de tiempo, es decir, si $y(k)$ es la salida correspondiente a la entrada $x(k)$ en un sistema invariante en el tiempo, la entrada $x(k-k_o)$ producirá la salida $y(k-k_o)$.

1.2.2.7. Linealidad

Un sistema lineal es aquel que posee la propiedad de superposición. Dicha propiedad se refiere a que si una entrada es la combinación lineal (suma ponderada) de varias señales, entonces la salida correspondiente es la combinación lineal de las salidas correspondientes a cada una de dichas entradas.

1.2.3. Señales discretas

Una señal de tiempo discreto, puede representar muestras de un fenómeno para el cual la variable independiente es en realidad continua. Por ejemplo, el procesamiento de voz por computadora digital, requiere representar la señal continua de voz por una secuencia discreta de valores que pueda ser procesado por un algoritmo de computadora.

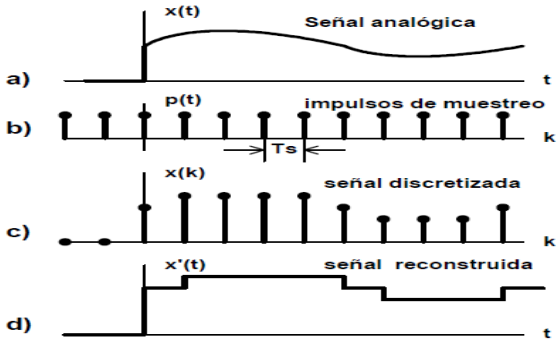
1.2.3.1. Proceso de muestreo

El proceso a través del cual una señal continua $x(t)$ es transformada en una señal discreta; es decir en $x(k)$ consiste simplemente en la toma de muestras de la señal continua en intervalos de tiempo k, denominados instantes de muestreo.

El proceso de muestro se observa en la figura 5. Para realizar dicho proceso es necesaria una señal adicional que marque el tiempo de la toma de muestras, idealmente dicha señal $p(t)$ es un tren de impulsos con una frecuencia $f_s = \frac{1}{T_s}$, denominada frecuencia de muestreo (en Hertz).

También es usual considerar dicha frecuencia en radianes por segundo $\omega_s = \frac{2\pi}{T_s}$. El muestreo puede ser uniforme (T_s constante) o no uniforme (T_s variable). A T_s se le llama también al período de muestreo.

Figura 5. **El proceso de muestreo**



Fuente: procesamiento de señales digitales. http://www.angelfire.com/falcon/shadowrsv /tsenales/curdsp1. Consulta: 10 de abril de 2013.

1.2.3.2. El teorema de Shannon

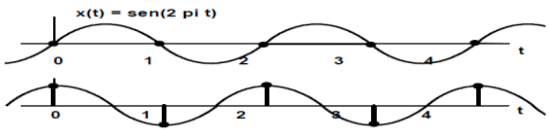
Se puede observar que en el proceso de muestreo al discretizar una señal de tiempo continuo se pierde algo de información en el proceso, es decir la información contenida $x(k)$ no es la misma que la de la señal original $x(t)$. Sin embargo, también se observa que $x(k)$ aún contiene algo de la información de $x(t)$.

Con las anteriores observaciones se puede realizar una pregunta: ¿será posible recuperar toda la información de una señal $\mathrm{x(t)}$, dada su versión discretizada $\mathrm{x(k)}$? El teorema de Shannon da la respuesta.

El teorema de Shannon prácticamente dice que entre más rápido se realice el muestreo, mejor representará $x(k)$ a la señal original $x(t)$, de manera que la condición para poder recuperar la información original deberá depender de la frecuencia de muestreo.

Para ilustrar esto, observe la figura 6, en la cual se está muestreando una señal continua a razón de 2 muestras por período, es decir: al doble de la frecuencia de la señal original.

Figura 6. **Señal continua muestreada a doble frecuencia**



Fuente: Procesamiento de señales digitales. http://www.angelfire.com/falcon/shadow_rsv/t_senales/curdsp1. Consulta: 10 de abril de 2013.

1.3. Herramientas básicas del procesamiento de señales

Ahora se verán las herramientas matemáticas con las cuales se pueden trabajar las señales en tiempo continuo y discreto, siendo estas la serie de Fourier en sus dos representaciones: tiempo continuo y tiempo discreto, así como la transformada de Fourier y la trasformada Z.

1.3.1. Serie y transformada de Fourier

El matemático francés Joseph Fourier, encontró que cualquier forma de onda periódica puede ser expresada por la suma de series de armónicas con relación senoidales, o senoidales cuyas frecuencias son múltiplos de la frecuencia fundamental de la señal en cuestión (o la primera armónica).

Por ejemplo una forma periódica puede expresarse de la siguiente manera siguiente:

$$f(t) = \frac{1}{2}a_0 + a_1\, cos wt + a_2\, cos 2wt + a_3\, cos 3wt + a_4\, cos 4wt + \cdots + b_1\, sin wt + b_2\, sin 2wt + b_3\, sin 3wt + b_4\, sin 4wt + \cdots \qquad \text{Ec. 1.1}$$

O abreviado:

$$f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}(a_n Cos\, nwt + b_n Sin\, nwt) \qquad \text{Ec. 1.2}$$

Donde el primer termino $\frac{1}{2}a_0$ es una constante, esta representa el valor promedio de f(t), y en una señal eléctrica es el valor DC. Los términos con los coeficientes a_1 y b_1 juntos representan los componentes w de la frecuencia fundamental.

De la misma manera los términos con coeficientes a_2 y b_2 representan los componentes debido a la segunda armónica 2w, y así sucesivamente.

Como cualquier señal periódica puede representarse como una serie de Fourier, se puede decir que las sumas de las componentes DC, primera armónica, segunda armónica, y así sucesivamente. Deben de producir la forma de onda f (t).

1.3.1.1. Coeficientes de Fourier

Los coeficientes de las series de Fourier se calculan con las siguientes integrales:

$$\frac{1}{2}a_0 = \frac{1}{2\pi}\int_0^{2\pi} f(t)dt \qquad \text{Ec. 1.3}$$

$$a_n = \frac{1}{2\pi}\int_0^{2\pi} f(t)\ cos\ nt\ dt \qquad \text{Ec. 1.4}$$

$$b_n = \frac{1}{2\pi}\int_0^{2\pi} f(t)\sin nt\ dt \qquad \text{Ec. 1.5}$$

Las ecuaciones anteriores combinadas con la ecuación 1.2, generan la forma de onda f(t), al agregar más coeficientes se tendrá una representación más exacta.

Las series de Fourier también suelen ser expresadas en su forma exponencial. La ventaja de usar la forma exponencial es que solo requiere una integración en vez de calcular dos para a_n y b_n . La forma exponencial se escribiría:

$$f(t) = \cdots + C_{-2}e^{-2jwt} + C_{-1}e^{-1jwt} + C_0 + C_1e^{jwt} + C_2e^{2wt} + \cdots \quad \text{Ecu. 1.6}$$

Donde el coeficiente Cn se obtiene de la siguiente integral:

$$C_n = \frac{1}{T}\int_0^T f(t)\, e^{jnwt}\, dwt \quad \text{Ec. 1.7}$$

También se puede obtener por relación de los coeficientes de la serie trigonométrica a_n y b_n.

$$a_n = C_{-n} + C_n \quad \text{Ec. 1.8}$$

$$b_n = j(C_n - C_{-n}) \quad \text{Ec. 1.9}$$

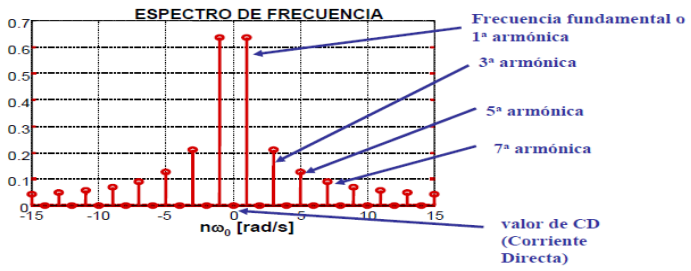
Una propiedad importante es que:

$$C_{-n} = C_n * \quad \text{Ec. 1.10}$$

Una vez que ya se conozca la serie de Fourier, es de mucha utilidad graficar las magnitudes de las armónicas de la forma de onda en la escala de la frecuencia.

A este tipo de gráfico se le conoce como líneas de espectro, y esta es la base de un analizador de espectro.

Figura 7. **Representación espectral**



Fuente: Señales. http://www.euv.cl/archivos_pdf/senales. Consulta: 5 de mayo de 2013.

En la figura 7 se muestra una representación de estas líneas de espectro. Se puede apreciar la ayuda que representa para reconocer las magnitudes debidas a cada armónica. En el caso de la figura anterior se indica que las magnitudes de las componentes de cada armónica van decreciendo.

Otro punto importante con las series de Fourier, es que presenta una forma sencilla de calcular el valor RMS (root mean square) de una señal. El valor RMS de una señal consta de senoidales de diferente frecuencia, es igual a la raíz cuadrada de la suma de los valores RMS de cada sinoidal al cuadrado.

1.3.1.2. Transformada de Fourier

Cuando se trabaja con series de Fourier, se debe notar que las señales deben ser periódicas en función del tiempo, estas formas de onda producían unas líneas de espectro discretas, localizadas en las frecuencias o armónicas de la forma de onda.

Ahora se tratará con señales no periódicas, como por ejemplo la función escalón unitario, función de rampa, o la delta entre otras. Se verá que estas funciones no periódicas van a producir una gráfica de espectro continuo.

Se supone una señal no periódica, se puede asumir que tiene un período extendiéndose desde -∞ a ∞, entonces para esta señal se formulá esta integral:

$$F(w) = \int_{-\infty}^{\infty} f(t)e^{-jwt}dt \qquad \text{Ec. 1.11}$$

La F(w) se llama la trasformada de Fourier o la integral de Fourier.

La función inversa de Fourier es la siguiente:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(w)e^{jwt}dw \qquad \text{Ec. 1.12}$$

Algunas propiedades importantes de la transformada de Fourier son:

- Linealidad**:** la propiedad de linealidad, establece que la transformada de Fourier de una suma de varias funciones es igual a la suma de las transformadas individuales.

- Simetría**:** si F(w) es la transformada de Fourier, entonces:

  $$F(t) \leftrightarrow 2\pi f(-w) \qquad \text{Ec. 1.13}$$

  O sea, si en F(w) se remplaza w por t, se obtendrá la relación de la ecuación 1.13.

- Escalamiento de tiempo**:** si *a* es una constante real, y si F(w) es la transformada de f(t):

$$\mathrm{f(at)} \leftrightarrow \frac{1}{|a|}\mathrm{F}\left(\frac{w}{a}\right) \qquad \text{Ec. 1.14}$$

Remplazando t por *at* en el dominio del tiempo, se tendrá que remplazar w por w/a en el dominio de la frecuencia y dividir F(w/a) por el valor absoluto de 1/a.

- Desplazamiento en el tiempo y frecuencia: si F(w) es la transformada de f(t):

$$f(t - t_0) \leftrightarrow F(w)e^{-jwto} \qquad \text{Ec. 1.15}$$

Diferenciación en el tiempo y en la frecuencia: siendo F(w) la transformada de f(t):

$$\frac{d^n}{dn^t} f(t) \leftrightarrow (jw)^n F(w) \qquad \text{Ec. 1.16}$$

Esta es la diferenciación en la frecuencia

$$(-jt)^n f(t) \leftrightarrow \frac{d^n}{dw^n} F(w) \qquad \text{Ec. 1.17}$$

- Convolución en el tiempo y frecuencia: si F1(w) es la transformada de Fourier de f1(t), y F2(w) es la respectiva de f2(t), entonces:

$$f1(t) * f2(t) \leftrightarrow F1(w)F2(w) \qquad \text{Ec. 1.18}$$

- Teorema de Parseval: si F(w) es la transformada de Fourier de f(t), entonces:

$$\int_{-\infty}^{\infty}|f(t)|^2 dt = \frac{1}{2\pi}\int_{-\infty}^{\infty}|F(w)|^2\, dw \qquad \text{Ecu. 1.19}$$

Esto es si la función f(t) representa el voltaje o corriente que pasa a través de una resistencia de 1 ohm, la potencia instantánea absorbida por el resistor es, v^2 o i^2. Entonces la integral de la magnitud al cuadrado, representa la energía (en watt por segundo o Joule) disipada por el resistor.

Por esta razón, la integral se le conoce como la energía de la señal. Según la relación dada en la ecuación 1.58, si por algún caso no se conoce la energía de la función del tiempo f(t), pero si se sabe su trasformada de Fourier, se puede calcular la energía sin la necesidad de evaluar la transformada inversa de Fourier.

1.3.2. Transformada discreta de Fourier (DTF)

El DFT es una secuencia en sí, en vez de una función con una variable constante y es correspondiente a las muestras espaciadas equitativamente en la frecuencia. La importancia de DFT es en la implementación de una variable de algoritmos del procesamiento digital de señales. Un punto importante que se debe notar, es que hay una relación entre secuencias periódicas y de duración finita.

Esta relación se obtiene construyendo una secuencia periódica, para la cual cada período es idéntico al de la secuencia de duración finita.

Es decir que la representación de las series de Fourier de una secuencia periódica corresponde a la DFT de una secuencia de duración finita. Para conseguir lo anterior, se debe conocer primero la representación de una secuencia periódica como la serie de Fourier, que también se conoce como series discretas de Fourier.

Considerando una secuencia $\tilde{x}[n]$ que es periódica con período N de la manera que $\tilde{x}[n] = \tilde{x}[n + rN]$ para cualquier valor n y r que sea entero, esta secuencia puede ser representada como una suma de armónicas relacionadas a secuencias exponenciales complejas. La representación de la serie de Fourier seria de la siguiente forma:

$$\tilde{x}[n] = \frac{1}{N}\sum_{K=0}^{N-1} \tilde{X}[k] e^{j\left(\frac{2\pi}{N}\right)kn} \qquad \text{Ec. 1.20}$$

Donde los coeficientes se encuentran con la siguiente ecuación:

$$\tilde{X}[k] = \sum_{K=0}^{N-1} \tilde{x}[k] e^{j\left(\frac{2\pi}{N}\right)kn} \qquad \text{Ec. 1.21}$$

Los coeficientes de la serie de Fourier pueden ser interpretados como la secuencia de duración finita, así como secuencia periódica.

Sin embargo una ventaja de interpretar los coeficientes $\tilde{X}[k]$ de la serie como secuencias periódicas, es que existe una dualidad entre el dominio del tiempo y de la frecuencia. El análisis anterior de la DFS es el camino hacia las transformadas de Fourier.

1.3.3. La transformada Z

Realiza la transformación del dominio de señales en tiempo discreto, a otro dominio llamado: dominio Z. Se utilizan las señales de tiempo discreto de la misma manera que las transformadas de Laplace y de Fourier usan señales de tiempo continuo.

La transformada Z conlleva una descripción de la frecuencia para las señales en tiempo discreto, y forma la base para el diseño de sistemas digitales, como los filtros digitales.

Una razón para introducir la transformada Z es que la transformada de Fourier no converge para todas las secuencias y es muy útil tener una generalización de la transformada de Fourier, que comprenda un mayor tipo de señales. Otra ventaja es que la notación es más conveniente en problemas analíticos.

La transformada de Fourier de una secuencia $X[n]$, se define como:

$$X\left(e^{jw}\right) = \sum_{n=-\infty}^{\infty} x[n]e^{-jwn} \qquad \text{Ec. 1.22}$$

La transformada Z de la misma secuencia es:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} \qquad \text{Ec. 1.23}$$

La ecuación es, en general, una suma infinita o una serie infinita de potencias, con Z siendo una variable compleja. El operador de la transformada se define como:

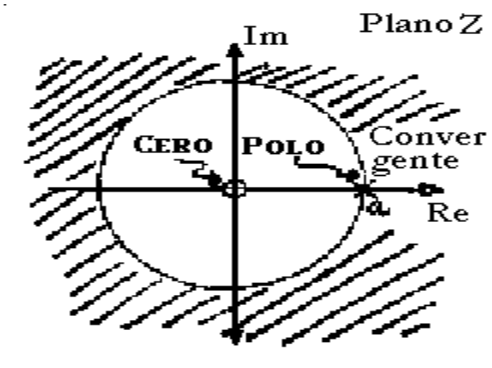
$$Z\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n]z^{-n} = X(z) \qquad \text{Ec. 1.24}$$

Con esta interpretación, se puede obtener el cambio del dominio de la secuencia $x[n]$. Otro que hay que resaltar es la relación que existe entre la transformada de Fourier y la transformada Z. Si se reemplaza la variable compleja z con la variable compleja e^{jw}, entonces la transformada Z sería la transformada de Fourier. Si se expresa la variable z en forma polar como $z = re^{jw}$, para el caso en que $|z| = 1$, la transformada z corresponde a la transformada de Fourier.

Para obtener una representación gráfica que ayude a describir y a interpretar la variable compleja, se usa el plano complejo z. En este plano el contorno responde a $|z| = 1$ que es el círculo unitario.

La siguiente gráfica muestra el plano complejo de Z:

Figura 8. **Plano complejo z**



Fuente: La trasformada z. http://dctrl.fib.unam.mx/ricardo/Transformada%20Z/La%20Transformada%20Z_corregido. Consulta: 15 de mayo de 2013.

La transformada Z es más útil cuando la suma infinita es expresada en una forma abreviada. Entre las más importantes y útiles transformadas Z se encuentran las que $X(z)$ es una función racional dentro de la región de convergencia.

$$X(z) = \frac{P(z)}{Q(z)} \qquad \text{Ec. 1.25}$$

Donde la $P(z)$ y $Q(z)$ con polinomios de z. Los valores para los cuales $X(z) = 0$ son llamados ceros y para los valores que hacen $X(z)$ sea infinito se llaman polos.

A continuación se presentan una tabla de los pares de transformadas Z.

Figura 9. **Pares de transformadas z**

	sucesión	transformada	región de convergencia
1.	$\delta[n]$	1	todo el plano z
2.	$\delta[n-m]$	z^{-m}	todo z salvo 0 (si $m>0$) o ∞ (si $m<0$)
3.	$u[n]$	$\frac{1}{1-z^{-1}}$	$\|z\|>1$
4.	$-u[-n-1]$	$\frac{1}{1-z^{-1}}$	$\|z\|<1$
5.	$a^n u[n]$	$\frac{1}{1-az^{-1}}$	$\|z\|>\|a\|$
6.	$-a^n u[-n-1]$	$\frac{1}{1-az^{-1}}$	$\|z\|<\|a\|$
13.	$\cos(\omega_0 n)\,u[n]$	$\frac{1-\cos\omega_0 z^{-1}}{1-2\cos\omega_0 z^{-1}+z^{-2}}$	$\|z\|>1$
14.	$\operatorname{sen}(\omega_0 n)\,u[n]$	$\frac{\operatorname{sen}\omega_0 z^{-1}}{1-2\cos\omega_0 z^{-1}+z^{-2}}$	$\|z\|>1$
15.	$r^n\cos(\omega_0 n)\,u[n]$	$\frac{1-r\cos\omega_0 z^{-1}}{1-2r\cos\omega_0 z^{-1}+r^2z^{-2}}$	$\|z\|>r$
16.	$r^n\operatorname{sen}(\omega_0 n)\,u[n]$	$\frac{r\operatorname{sen}\omega_0 z^{-1}}{1-2r\cos\omega_0 z^{-1}+r^2z^{-2}}$	$\|z\|>r$

Fuente: La transformada z. http://dctrl.fi-b.unam.mx/ricardo/Transformada%20Z/La%20Transformada%20Z_corregido. Consulta: 15 de mayo de 2013.

1.4. Conversión Analógica-Digital (ADC)

Muchos equipos y dispositivos modernos requieren procesar las señales analógicas que reciben y convertirlas en señales digitales para poder funcionar, es por ello la importancia que se le da a este tema en este trabajo, ya que es importante tener claro los procesos que conlleva la conversión Analoga-Digital.

Para explicar este tema el conversor ADC (*Analog*-to-*Digital converter*-conversor Analogico Digital), tiene que efectuar los siguientes procesos:

- Muestreo de la señal analógica
- Cuantizacion de la propia señal
- Codificación del resultado de la cuantizacion en código binaria.

1.4.1. Muestreo de la señal analógica

Para convertir una señal analógica en digital, el primer paso consiste en realizar un muestreo (*sampling*) de esta, o lo que es igual, tomar diferentes muestras de tensiones o voltajes en diferentes puntos de la onda senoidal. La frecuencia a la que se realiza el muestreo se denomina razón, tasa o también frecuencia de muestreo y se mide en kilohertz (kHz).

1.4.2. Cuantizacion de la señal analógica

Una vez realizado el muestreo, el siguiente paso es la cuantización de la señal analógica. Para esta parte del proceso los valores continuos de la sinusoide se convierte en series de valores numéricos decimales discretos, correspondientes a los diferentes niveles o variaciones de voltajes que contiene la señal analógica original.

Por lo tanto, la cuantizacion representa el componente de muestreo de las variaciones de valores de tensiones o voltajes tomados en diferentes puntos de la onda sinusoidal, que permite medirlos y asignarles sus correspondientes valores en el sistema numérico decimal, antes de convertir esos valores en sistema numérico binario.

1.4.3. Codificación de la señal en código binario

Después de realizada la cuantizacion, los valores de la toma de voltajes se representan numéricamente por medio de códigos y estándares previamente establecidos.

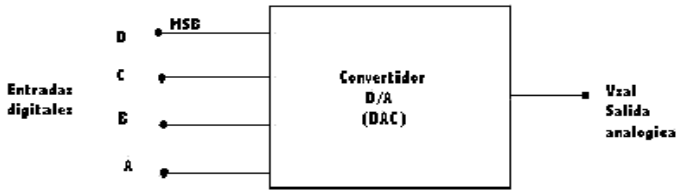
Lo más común es codificar la señal digital en código numérico binario.

1.4.4. Conversión Digital-Analógica (DAC)

Este proceso es el inverso al visto anteriormente, en donde de muestras digitales (valores discretos) se convierten en señales analógicas (valores continuos).El conversor digital analógico asocia cada valor binario a un nivel de tensión establecido y genera muestras de tensión utilizando dichos niveles, aplicando un intervalo de tiempo constante entre muestras.

Un conversor digital es un dispositivo que recibe la información en forma de palabra de n-bits y la transforma en una señal analógica, la señal obtenida no es una señal continua sino que se obtiene un número discreto de escalones como consecuencia de la discretizacion de la entrada. En la figura 10 se puede ver un diagrama general de lo que es un convertidor D/A.

Figura 10. **Diagrama convertidor DAC**



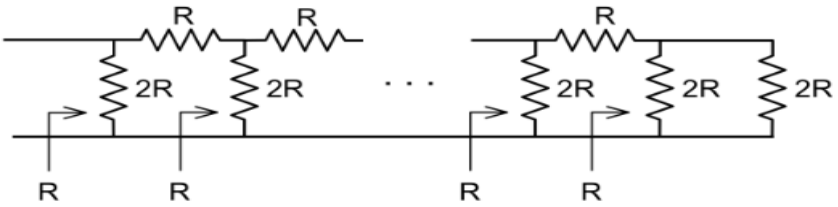
Fuente: Convertidor DAC. http://proton.ucting.udg.mx/~sanchez/EL%20DAC.htm. Consulta: 2 de junio de 2013

La conversión se realiza por la suma ponderada de los dígitos de valor 1, se consigue de una forma muy simple, un conversor digital analógico, esta ponderación se puede realizar con un método sencillo que consta de una serie de resistencias en progresión geométrica, es decir cada una mitad de la anterior, lo cual obliga a utilizar un amplio rango de resistencias o bien mediante una red R-2R que efectúa sucesivas divisiones por 2.

1.4.4.1. Método R-2R

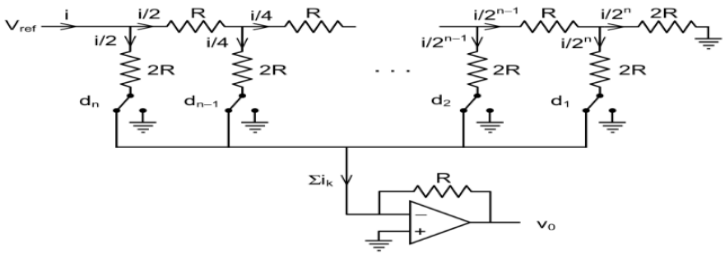
Una red resistiva como la que se puede observar en la figura 11, tiene la particularidad de que cualquiera sea el número de secciones la resistencia vista (excepto la final) es R. Este circuito puede usarse como se muestra en la figura 12, siguiente para obtener un conversor sencillo, pero eficiente y será el método que se utilice en una práctica de este trabajo.

Figura 11. **Una red R-2R**



Fuente: Conversores D/A y A/D. http://www.fceia.unr.edu.ar/enica3/da-ad. Consulta: 16 de junio de 2013.

Figura 12. **Conversor analógico-digital R-2R**



Fuente: Conversores D/A y A/D. http://www.fceia.unr.edu.ar/enica3/da-ad. Consulta: 16 de junio de 2013

La ecuación que rige para este método es la siguiente figura:

Figura 13. **Ecuación método R-2R**

$$v_o = -\frac{V_{ref}}{R} R \sum_{k=1}^{n} \frac{d_k}{2^{n-k+1}} = -\frac{V_{ref}}{2^n} \sum_{k=1}^{n} d_k 2^{k-1},$$

Fuente: Conversores D/A y A/D. http://www.fceia.unr.edu.ar/enica3/da-ad. Consulta: 16 de junio de 2013.

En el capítulo 5 se realizará la práctica utilizando un dsPIC30F4013 y red R-2R de 8 bits.

1.5. Filtros digitales

Es un algoritmo implementado en hardware o software como será el caso de este trabajo que se verá las 2 maneras, dicho algoritmo es aplicado a una señal de entrada digital (discreta en tiempo y cuantizada en amplitud) y genera una señal digital de salida luego de haber sido efectuado el proceso de filtrado.

En la figura 14 se muestra un diagrama de bloques simplificado de un filtro digital que opera en tiempo real con entradas y salidas analógicas.

Figura 14. **Diagrama de bloques de un filtro digital**

Fuente: elaboración propia, con programa Microsoft Word 2010.

Los filtros juegan un papel de gran importancia en el procesamiento digital de señales, entre sus aplicaciones se tienen:

- Comprensión de datos
- Procesamiento de señales biomédicas
- Procesamiento de señales de voz
- Procesamiento de imágenes
- Trasmisión de datos

- Cancelación de ecos telefónicos

La operación de filtrado se realiza por medio de cálculos directos con las señales muestreadas. Las ventajas que representan los filtros digitales frente a los analógicos, son las siguientes:

- Respuesta dinámica: el ancho de banda del filtro digital está limitado por la frecuencia de muestreo, mientras que en los filtros analógicos con componentes activos suelen estar restringidos por los amplificadores operacionales.
- Intervalo dinámico: en filtros analógicos aparecen parámetros que limitan por abajo el rango y se saturan con la alimentación. En cambio en los filtros digitales es fijado por el número de bits que representa la secuencia, y el límite inferior por el ruido de cuantificación y los errores de redondeo.
- Conmutabilidad: si los parámetros de un filtro se conservan en registros, los contenidos de dichos registros pueden ser modificados a voluntad. Además estos filtros se pueden conmutar, pudiéndose multiplexar en el tiempo para procesar varias entradas a la vez.
- Adaptabilidad: un filtro digital puede ser implementado en soporte físico (hardware) o mediante de un programa de computadora (software).
- Ausencia de problemas de componentes: los parámetros de los filtros se representan por medio de números binarios y no derivan con el tiempo.

Al no haber componentes, no hay problemas de tolerancia o deriva de componentes, y ningún otro problema asociado con un comportamiento no ideal de resistencias, condensadores, bobinas o amplificadores. Tampoco existen problemas de impedancia de entrada ni salida, ni efectos de adaptación de impedancias entre etapas.

Una distinción fundamental en los sistemas discretos dinámicos lineales e invariantes, y en particular en los filtros digitales, es la duración de la respuesta ante el impulso. Existen sistemas de repuesta de pulso finito o no recursivo (FIR, *finite impulse response*), y de sistemas de repuesta infinita o recursivo (IIR, *infinite impulse response*).

Partiendo de la ecuación en diferencias que modela el comportamiento dinámico de estos sistemas, se tiene:

$$Y_k = a_1 y_{k-1} + a_s y_{k-2} + \cdots + a_n y_{k-n} + b_0 x_{k-0} + b_1 x_{k-1} + \cdots + b_m x_{k-m} \quad \text{Ec. 1.26}$$

En el caso de tener todos los coeficientes a_i iguales a cero, se tendrá un filtro FIR, con lo quedara la ecuación reducida a:

$$Y_k = b_0 x_{k-0} + b_1 x_{k-1} + \cdots + b_m x_{k-m} \quad \text{Ec. 1.27}$$

En estos filtros el valor de la secuencia de salida, solo depende de un número finito de valores de la secuencia de entrada. Además también se desprende la carencia de polos en la función de trasferencia. En cambio, las expresiones de los filtros recursivos corresponden a:

$$Y_k = a_1 y_{k-1} + a_s y_{k-2} + \cdots + a_n y_{k-n} + b_0 x_{k-0} + b_1 x_{k-1} + \cdots + b_m x_{k-m} \quad \text{Ec. 1.28}$$

La transformada Z de la ecuación 1.29 es:

$$G(z) = \frac{b_0 + b_a z^{-1} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} \quad \text{Ec.1.29}$$

En estos casos, la secuencia de salida depende tanto de la entrada como de la salida. De estas ecuaciones se deducen las siguientes propiedades; primero la secuencia de ponderación es infinita para los filtros IIR, aún teniendo un número finito de coeficientes, mientras la respuesta al impulso de un filtro no recursivo es siempre finita e igual al orden del filtro.

En segundo lugar, los filtros FIR son siempre estables, esto es, la secuencia de salida tiene todos sus valores acotados. No es el caso de los filtros recursivos, su estabilidad depende de la función de transferencia, por lo que habrá de utilizar alguno de los procedimientos algebraicos, para analizar su estabilidad.

Tercera, cualquier filtro recursivo puede ser reemplazado por otro no recursivo con infinitos coeficientes, sus valores vendrán dados por la secuencia de ponderación del IIR. De manera inversa no se cumple.

1.5.1. Filtros no recursivos (FIR)

Los filtros no recursivos tienen ventajas muy interesantes que les hacen ser ampliamente utilizados en múltiples aplicaciones.

La característica más destacable es su facilidad de diseño para conseguir una respuesta en frecuencia en fase lineal, esto es, la señal que pase a través de él no será distorsionada. Los FIR son por su propia constitución estables, no habiendo problemas en su diseño o fase de implementación.

Su mayor desventaja está en que para cumplir con las especificaciones dadas, el filtro FIR necesita un orden mayor al que se obtuviera con un IIR, implicando programas más largos o circuitos mayores.

Los filtros digitales suelen ser caracterizados en términos de rangos de frecuencia, tanto de la banda pasante como de la supresora. Sus respuestas en la frecuencia son periódicas con la frecuencia de Nyquist, w_N , por lo que solo se considera el intervalo $[-w_N, w_N]$.

El modelo matemático que caracteriza la respuesta en frecuencia de un filtro típico pasa bajo es:

$$G(w) = \begin{cases} e^{-j\lambda w} & |w| \leq w_c \\ 0 \; En\ otro\ caso \end{cases} \qquad \text{Ec.1.30}$$

La notación w_c indica la frecuencia de corte, donde λ es el desfase, y w_s la frecuencia de muestreo. Si se aplica la transformada inversa a la ecuación 1.30 se obtiene:

$$g_n = \frac{1}{2\pi}\int_{-w_c}^{w_c} G(w)e^{jnwt}dw = \frac{2w_c}{w_s}\frac{\sin[(n-\lambda)w_cT]}{(n-\lambda)w_cT}\ para\ n = \cdots -2, -1, 0, 1, 2 \ldots \text{Ec. 1.31}$$

Una respuesta en frecuencia del filtro se representa de la siguiente manera:

$$G(w) = e^{-jnwT}\{g_N + 2\sum_{i=0}^{N-1} g_i \cos[(N-i)wT]\} \qquad \text{Ec.1.32}$$

El desfase introducción por el filtro es –NwT.

Este mismo análisis se puede utilizar para diseñar otro tipo de filtros, únicamente es necesario utilizar una transformación como se muestra en la siguiente figura.

Figura 15. **Conversión de filtros digitales**

Conversión	Transformación	Parámetros
A paso alto	$g_{n(PA)} = (-1)^n g_{n(PB)}$	$(w_c)_{PA} = (w_N) - (w_c)_{PB}$
A pasa banda	$g_{n(PBANDA)} = (2\cos(nw_0T))g_{n(PB)}$	$w_0 = frecuencia\ central$ $w_1 = w_0 - (w_c)_{PB}$ $w_2 = (w_c)_{PB} - w_0$

Fuente: Diseño de filtros digitales. http://www.ingelec.uns.edu.ar/pds2803/ Materiales/Cap07/07-Cap07. Consulta: 27 de junio de 2013.

Algo que hay que mencionar es que, aunque el orden del filtro sea elevado, los rizados tanto en la banda pasante como en la supresora se mantienen, haciéndose mayores las oscilaciones en las zonas de transición entre las bandas. Además, la atenuación en la banda no pasante no es cero y la transición entre las bandas no es abrupta. A este fenómeno se llama efecto de Gibbs.

Sin embargo, la aplicación de ciertas funciones ventanas, permiten aliviar este efecto no deseado. Si se quiere disminuir las oscilaciones, la respuesta del pulso infinito original debe ser multiplicada por una función ventana que sea un pulso rectangular pura, de manera que la nueva secuencia de ponderación gn' quedará como:

$$g'_n = g_n w_n \qquad \text{Ec. 1.33}$$

Donde g_n es la función de la ecuación 1.32 y w_n una función ventana definida por:

$$w_n = \begin{cases} 1 & |n| \leq N \\ 0 & |n| > N \end{cases} \qquad \text{Ec.1.34}$$

En el cual N es el número de la secuencia de transición entre la banda pasante y la supresora, aunque en la práctica esta ventana causaría también el efecto Gibbs. Para evitarlo hay diferentes tipos de ventanas, como la que sigue:

$$w_n = \begin{cases} \propto +(1-\propto)\cos\left(\frac{\pi n}{N}\right) & |n| \leq N \\ 0 & |n| > N \end{cases} \qquad \text{Ec. 1.35}$$

En la cual si $\propto = 0{,}5$, se le conoce como ventana de von Hann, y cuando $\alpha = 0{,}54$, se denomina ventana de Hamming.

La ventana de Blackman se define como:

$$w_n = \begin{cases} 0{,}42 + 0{,}5\cos\left(\frac{\pi n}{N}\right) + 0{,}08\cos\left(\frac{2\pi n}{N}\right) & |n| \leq N \\ 0 & |n| > N \end{cases} \qquad \text{Ec.1.36}$$

Las características de esta ventana es que reduce el rizado en comparación con las dos anteriores, pero la transición entre bandas es muy suave. Si lo que se busca es una relación entre el rizado y la ruptura abrupta, es preferible usar una ventana de Kaiser definina como:

$$w_n = \begin{cases} \frac{I_0(\beta)}{I_0(\alpha)} & |n| \leq N \\ 0 & |n| > N \end{cases} \qquad \text{Ec. 1.37}$$

Siendo α un parámetro y

$$\beta = \alpha\sqrt{(1 - \left(\frac{n}{N}\right)^2)} \qquad \text{Ec. 1.38}$$

Donde $I(x)$ es una función de Bessel de orden cero definida por la serie:

$$I_0(x) = 1 + \sum_{K=1}^{\infty}(\frac{1}{k!}\left(\frac{x}{2}\right)^k)^2 \qquad \text{Ec. 1.39}$$

A medida de tener β se disminuye el rizado, pero también disminuye la pendiente de transición entre la banda pasante y la supresora.

1.5.2. Filtros recursivos (IIR)

La ventaja de los filtros IIR respecto a los FIR, es la de tener un menor orden del filtro para iguales especificaciones de diseño. Aunque la desventaja, es la falta de desfase lineal introducido por el filtro, asi como la necesidad de realizar estudios de estabilidad, pues esta no está garantizada en el diseño.

Los filtros no recursivos basados en modelos conocidos, como por ejemplo:

Butterworth, Chebyschev, entre otros, pueden se diseñados por varios métodos, siendo el más común el basado en las transformaciones bilineales. Este procedimiento requiere del conocimiento de la función de transferencia en el dominio s del filtro a diseñar.

Los coeficientes del filtro en el dominio s son transformados a uno equivalente en el dominio z, y los coeficientes discretos formaran el filtro IIR en tiempo discreto.

El método de diseño de filtros IIR basados en transformaciones bilineales, tiene el siguiente procedimiento.

- Paso 1: definir las características del filtro digital $w_{d1}, w_{d2}, \dots, w_{dk}$

- Paso 2: realizar la operación de la siguiente función:

$$w_{ai} = \frac{2}{T}\tan\left(\frac{w_{d1}T}{2}\right) \quad 1 \leq i \leq K, \qquad \text{Ec. 1.40}$$

De esta manera se encuentran las frecuencias analógicas

- Paso 3: cambiar por S en el filtro analógico los resultados de:

$$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} = \frac{2}{T}\frac{z-1}{z+1} \qquad \text{Ec. 1.41}$$

2. SOFTWARE APLICADO AL PROCESAMIENTO DIGITAL DE SEÑALES

En el capítulo anterior se dio a conocer toda la teoría respecto al procesamiento digital así como su matemática, ahora se describe un software de mucha utilidad para este trabajo, que es MATLAB de la empresa Mathworks, y será en este programa donde se realizaran 3 prácticas, estas serán presentadas en el capítulo 4, pero antes se dará a conocer este sorprendente software y describir también los comandos necesarios para poderlo utilizar de una manera adecuada.

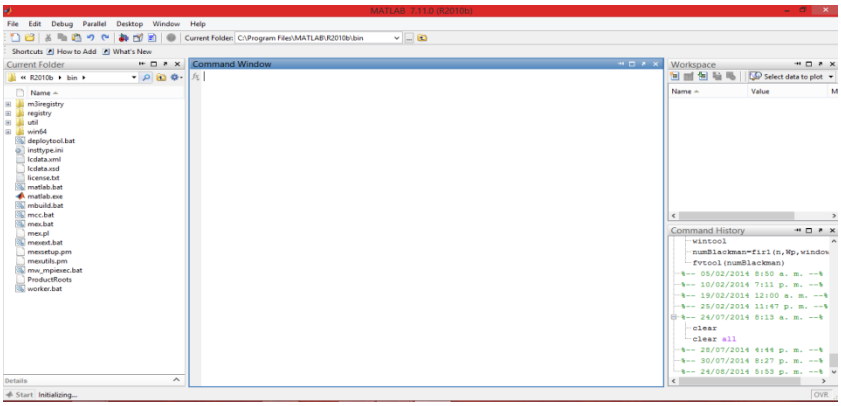
2.1. ¿Qué es MATLAB?

MATLAB es el nombre abreviado de MATris LABoratory. MATLAB es un programa para realizar cálculos con vectores y matrices, puede también trabajar con números escalares, tanto reales como complejos.

En MATLAB se pueden realizar cálculos sencillos como lo son sumas, restas, multiplicaciones y otros. Hasta el manejo de filtros, transformada de Fourier, procesamiento de imágenes, procesamiento de voz.

El entorno de trabajo de MATLAB es muy gráfico e intuitivo, similar a muchas ventanas que se manejan en Windows. El espacio de trabajo (Workspace) es un conjunto de variables y de funciones de usuario, que en un determinado momento están definidas en la memoria del programa o de la función que se está ejecutando. La ventana Workspace constituye un entorno gráfico para ver las variables definidas en el espacio de trabajo. Ver figura 16.

Figura 16. **Entorno de trabajo de MATLAB R2010b**



Fuente: MATLAB R2010b

2.1.1. El editor de MATLAB

En MATLAB tienen particular importancia los archivos-M o M-files. Son ficheros de texto ASCII con la extensión *.m* que contiene conjunto de comando o definición de funciones. La importancia de estos ficheros-M es que al teclear su nombre en la línea de comandos, se ejecutan todos los comandos contenidos en dicho fichero, uno tras otro.

Aunque los ficheros *.m* se pueden crear con cualquier editor de ficheros ASCII tal como *notepad*, MATLAB dispone de un editor que permite tanto crear y modificar estos ficheros, como ejecutarlos paso a paso para ver si contienen errores a este proceso se le llama *debug* o depuración.

2.1.2. Aplicaciones de MATLAB

MATLAB en la actualidad es utilizado para diversas aplicaciones, ya que cuenta con una gran cantidad de herramientas llamadas *toolbox,* que se utilizan en diversos campos de electrónica y en otros campos, algunas de sus aplicaciones son:

- Matemática
- Física
- Estadística
- Finanzas
- Sistemas de control
- Biomédica
- Electrónica de potencia
- Procesamiento de señales

Pero para este trabajo el área en la que se enfocará será para el procesamiento de señales.

2.2. MATLAB y sus aplicaciones en el tratamiento digital de señales

MATLAB es una herramienta de mucha utilidad para el análisis y procesamiento de señales, ya que cuenta con varios comandos para este tipo de aplicaciones, aquí se muestra y describe los comandos utilizados a lo largo de las prácticas que se encuentran en el capítulo 4.

2.2.1. Generación de señales

En MATLAB las señales se representan ya sea como vectores o como matrices, para generar una señal se puede especificar cada elemento de la señal o utilizar las funciones matemáticas estándar. Las señales más utilizadas son sinusoidales o aleatorias.

2.2.2. Procesamiento de audio

MATLAB cuenta un conjunto de comandos y funciones para el procesamiento de audio, que van desde como reproducir un sonido, grabar uno utilizando el micrófono de la computadora o un externo y guardar sonidos en una determinada variable, los comandos más importantes para este trabajo, son los siguientes:

- Soundsc(xt,Fs): sirve para escuchar un tono
- r = audiorecorder: crea un objeto de grabación
- Pause(r): pausa el archivo de audio
- Stop(r): finaliza el archivo de audio
- Play(r): escuchar la grabación
- y=getaudiodata(r): para obtener la matriz que contiene las muestras de la señal audible. Esta señal se puede procesar
- Fs=r.samplerate: para obtener la frecuencia de muestreo
- Wavwrite(y,Fs,´grabacion´): para guardar en formato wav en la carpeta de trabajo de MATLAB
- [xt,Fs]=wavread('nombre_de_archivo'): leer un archivo de audio en formato wav que se encuentra en la carpeta de trabajo

2.2.3. Análisis de Fourier

MATLAB está equipado con funciones especiales que van a permitir realizar un análisis de Fourier, de funciones definidas por un conjunto de valores discretos. Por ejemplo el comando *fft()* permite obtener la transformada rápida de Fourier (*fast Fourier Transform*) de una secuencia de números definida por el vector x.

Los comandos más utilizados son los siguientes:

- abs: da el valor absoluto y la magnitud compleja.
- angle: da el angulo de fase.
- fft: para obtener la transformada rapida de Fourier.
- fft2: para obtener la transformada rapida de Fourier en 2 dimensiones.
- ifft: para obtener la transformada inversa de Fourier.
- fftshift: reordena el verctor x en orden creciente de frecuencia
- ifft2: para obtener la transformada inversa de Fourier en 2 dimensiones.

Otros comandos importantes en este tema, son los de convulción que son los siguientes:

- conv: convolución y multiplicación de polinomios.
- conv2: convolución en 2 dimenciones.

2.2.4. Diseño de filtros digitales

En esta sección se dará a conocer los comandos básicos para el diseño de filtros en MATLAB, ya que este conocimiento será de mucha ayuda para realizar las prácticas en el capítulo 4.

Después de tener el conocimiento teórico adquirido en el capítulo 1, ahora se describe como diseñar filtros con MATLAB.

2.2.4.1. Filtros IIR

Hay varias funciones y maneras de calcular los coeficientes de un filtro IIR, en MATLAB. Se tienen herramientas que calculan los parámetros apropiados según la aplicación y otras que usan esta información para calcular los coeficientes.

A continuación se describe la estructura de cómo obtener los coeficientes de este tipo de filtros:

[N;Wn]= función (Wp, Ws, Rp, Rs)

Donde:

Wp: frecuencia de pasabanda
Ws: frecuencia de rechazabanda
Rp: máxima atenuación permitida en pasabanda
Rs: mínima atenuación deseada en rechazabanda
N: orden mínimo calculado para el filtro.
Wn: frecuencia natural. Parámetro de entrada necesario para que el filtro calculado mediante la función correspondiente cumpla con las especificaciones del diseño.

Las funciones que se pueden utilizar son:

- [N;Wn]= buttord(Wp, Ws, Rp, Rs)

- [N;Wn]= cheb1ord(Wp, Ws, Rp, Rs)
- [N;Wn]= cheb2ord(Wp, Ws, Rp, Rs)
- [N;Wn]= ellipord(Wp, Ws, Rp, Rs)

Para el diseño de filtros Butterworth pasa bajos y pasa altos se tienen las instrucciones siguientes:

- [B, A] = butter(N, Wn); y [B, A] = butter(N,Wn, ´high´);

Que almacenan el B y en A los N+1 coeficientes para la implementación de la función de transferencia deseada. Si Wn = [W1 W2] con 0<W1<W2<1, butter calcula los coeficientes para un pasabanda y rechasabanda mediante los comandos:

- [B, A] = butter(N, Wn); y [B, A] = butter(N,Wn, ´stop´);

El funcionamiento y los parámetros de las instrucciones para los filtros Chebischev tipos I y II son muy similares a los de butter.

- [B, A] = cheby1(N, R, Wn); y [B, A] = cheby2(N, R, Wn);

Una vez que se tiene los parámetros N y Wn del filtro Elíptico, se agrega la información del riple de pasabanda Rp y el de rechazabanda Rs, según el siguiente formato:

- [B, A] = ellipord(N, Rp, Rs Wn);

2.2.4.2. Filtros FIR

MATLAB cuenta con varios comandos y formas de poder diseñar filtros FIR, en esta sección se describen los comandos más importantes y en la práctica del capítulo 4 se ampliará este tema.

Los pasos más importantes en el diseño de filtros FIR en MATLAB son:

- Escoger una respuesta ideal, normalmente en el dominio de la frecuencia.
- Escoger un tipo particular de filtro.
- Escoger un criterio de medida para valorar la aproximación.
- Definir un método para seleccionar la mejor aproximación.

Estos pasos son repetitivos hasta encontrar la especificación del filtro que se ajusta a la necesidad del diseño. Es posible que en esta etapa del diseño se quiera redefinir la respuesta ideal, el tipo de filtro o el criterio de medida. De cualquier manera, MATLAB ofrece un conjunto de funciones y rutinas que permiten agilizar este proceso.

La función *fir1* ofrece una rutina interactiva para el diseño de filtros FIR pasabajo y pasabanda. Los argumentos de esta función son fundamentalmente; el orden del filtro N y la frecuencia normalizada de corte *Wn* definida con el criterio *0<Wn<1,* donde uno corresponde a la mitad de la frecuencia de muestreo. Su formato es:

- B=FIR1(N,Wn,type,window);
- B=FIR1(N, Wn). Calcula y almacena en B N+1 coeficientes de u filtro FIR pasa bajos.

- B=FIR1(N, Wn, ‘hing’) usa el parámetro ‘high’ para diseñar filtros FIR pasa altos.

Al momento de diseñar filtros pasa banda o rechaza banda con la función *fir1,* se debe definir *Wn* como un valor de dos elementos *Wn= [W1 W2]*, con la restricción *0<W1<W<W2<1*

- B=FIR1(N, Wn). Calcula y almacena en B los N+1 coeficientes de un filtro FIR pasabanda.

- B=FIR1(N, Wn, ´stop´); usa el parámetro ´stop´ para diseñar filtros FIR rechazabanda.

Se usa por *default* la ventana Hamming. Sin embargo, se puede usar ventanas rectangulares, Hamming, Blackman, Kaiser and Chebyshev. El comando se escribe de la siguiente manera:

- B=FIR1(N, Wn, rectangular(N+1));

2.3. Compilador *mikrobasic for dsPIC*

El compilador *mikrobasic* es un software desarrollado por la empresa MikroElectronica, el cual viene con la placa entrenadora EasydsPIC4A o se puede descargar una versión sencilla desde su página web. Este software consta de una amplia variedad de librerías que permiten una programación efectiva y fácil acceso al usuario.

Las características más importantes que tiene este compilador son:

- Permite escribir programas en lenguaje Basic.
- Utilización de librerías incluidas para el mejoramiento, rapidez de adquisición de datos, memoria, conversiones y comunicaciones.
- Monitorear la estructura del programa, variables, y funciones utilizando un explorador de código.
- Un depurador para inspeccionar el flujo del programa paso a paso.
- Mapas de visualización de memoria ROM y RAM.
- Amplia variedad de modelos de microcontroladores dsPIC.
- Generación del archivo ensamblador (ASM) y hexadecimal (HEX) estándar compatible para cargar el código a la memoria del microcontrolador.

Lo primero que se realiza al iniciar el compilador es crear un proyecto que contendrá todas las especificaciones y parámetros como el modelo de microcontrolador dsPIC a utilizar y la frecuencia de operación del oscilador, librería de funciones entre otros. El proceso de crear y ejecutar un proyecto en mikrobasic es el siguiente:

- Crear un proyecto (nombre de proyecto, configuración de proyecto, dependencia entre archivos).
- Editar un programa.
- Depurar el código, ejecutando el programa paso a paso para asegurarse de que se ejecuten las operaciones deseadas.
- Copilar el programa y corrección de errores.
- Programar un microcontrolador (cargar el archivo HEX generado por el compilador en el microcontrolador utilizando el programador dsPICflash).

2.3.1. Carga del código al dsPIC

El programador dsPICflash es una herramienta diseñada para programar todo los tipos de microcontroladores dsPIC, está compuesto de dos partes.

La parte de hardware se utiliza para poner en el *bufer* el código hexadecimal y para programar el microcontrolador por medio de niveles de voltaje, durante este proceso un nuevo programa se escribe en la memoria *flash* del microcontrolador, mientras que el programa anterior es borrado.

La parte de software se encarga de enviar el programa (archivo HEX) a la parte de hardware del programador por medio de un cable USB a la placa entrenadora EasydsPIC4A. Antes de enviar el código es posible modificar algunas configuraciones del programador y controlar el funcionamiento de la parte hardware como cargar un archivo, escribir el código en la memoria del PIC, limpiar la memoria entre otras.

3. ESTUDIO Y ANÁLISIS DEL MICROCONTROLADOR DE PROCESAMIENTO DIGITAL DSPIC

La electrónica ha avanzado rápidamente en los últimos años, especialmente en la técnica de fabricación de circuitos integrados y esto sin duda seguirá así, teniendo un gran impacto en diferentes áreas de la industria y sociedad.

El rápido desarrollo de la tecnología en circuitos electrónicos a estimulado el desarrollo de computadoras digitales más potentes, pequeños, rápidos y baratos y de hardware digital de propósito general. Por eso en este capítulo se describe un circuito integrado que está teniendo bastante auge en la industria de la electrónica.

En este capítulo se trata de guiar al estudiante hacia uno de los temas principales de este trabajo, que es el DSPIC, se da a conocer sus características principales y un análisis breve de cada una de ellas.

3.1. Definición y características del DSP

Procesador Digital de Señales (DSP) es un circuito integrado que contiene un procesador y una serie de recursos capaces de manejar y manipular las señales analógicas en tiempo real, como son los sonidos y las imágenes.

En la década de los 80 ya se comercializaban varios modelos de DSP, siendo pioneros los fabricantes Texas Instrumets, NEC e Intel. En la actualidad Texas Intruments, fabrica modelos de DSP más potentes para realizar aplicaciones cada vez más complejas.

Los DSP son muy similares a los convencionales microcontroladores, con la diferencia que estos incorporan arquitecturas y recursos especiales para poder trabajar de una forma óptima los requerimientos en el procesamiento de señales analógicas.

De sus características se puede destacar lo siguiente:

- Comúnmente los procesadores son RISC, con un reducido conjunto de instrucciones que se ejecutan generalmente en un ciclo.
- Cuentan con una arquitectura Harvard y disponen de dos memorias independientes, una dedicada para las instrucciones y la otra para los datos, lo cual permite el acceso simultáneo a ambas informaciones.
- Los modos de direccionamiento son muy sofisticados, ya que localizan los datos y almacenan los resultados de forma eficiente según lo requiera el algoritmo.
- Disponen de un conjunto de interrupciones muy amplio y veloz con niveles de prioridad.
- Integra un conjunto de recursos y periféricos que optimizan el tamaño y simplifican el diseño.
- Poseen módulos para la optimización de energía.

3.1.1. Arquitectura Harvard

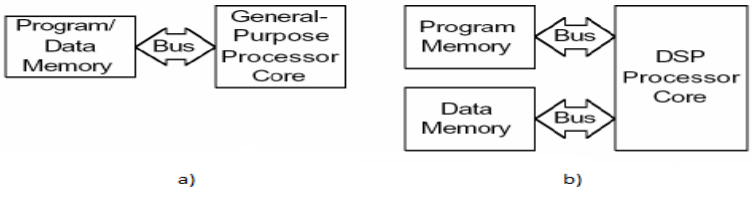
En la arquitectura clásica de Neumann la ALU y la unidad de control están conectadas a una solo unidad de memoria que almacena tanto instrucciones de programa como datos. Durante la ejecución de un programa, la instrucción es leída desde la memoria y decodificada, los operando necesarios son obtenidos desde la memoria y finalmente la instrucción es ejecutada.

La desventaja principal es que la memoria se transforma en cuello de botella de esta arquitectura.

En cambio las instrucciones que más ejecuta un DSP estándar es la multiplicación y acumulación, esta debe ser realizada con eficiencia. Es por eso que la arquitectura Harvard cuenta con 2 memorias; una de ellas es utilizada exclusivamente para datos, mientras que la otra es utilizada para instrucciones.

Esta arquitectura alcanza un alto grado de concurrencia (lecturas y escrituras simultáneamente).

Figura 17. **a) Arquitectura von Neumann b) Arquitectura Harvard**



Fuente: elaboración propia, con programa Microsoft Word 2010.

3.1.2. Diferencia entre microcontrolador y DSP

Un microcontrolador está formado por circuitos integrados que contiene un procesador digital completo junto a diversos periféricos auxiliares. La diferencia con respecto al DSP es muy grande, por ello sus campos de aplicación son muy diferentes.

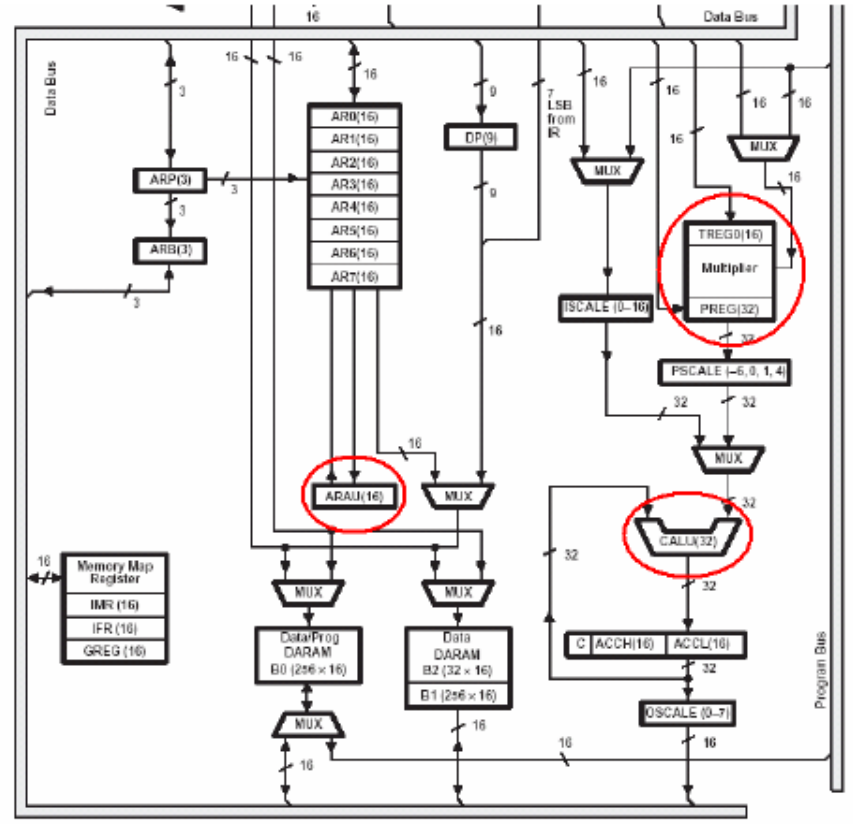
En las instrucciones aritméticas complejas el microcontrolador lo realiza en varios ciclos de reloj, mientras que los DSP lo ejecutan en un solo ciclo. Los DSP cuentan con A/D rápidos y precisos.

Dado el carácter matemático de los programas para el DSP, estos están preparados para ser programados en lenguajes de alto nivel.

3.1.3. Diagrama de bloques de un DSP

En la figura 18 se puede observar un diagrama de bloques de un DSP, en este caso es el del modelo TMS320F241. Se pueden ver tres unidades de cálculo, CALU, ARAU, y una unidad de multiplicación, la cual permite además realizar corrimientos.

Figura 18. **Diagrama de bloques del DSP TMS320F241**



Fuente: Texas Instruments. DSP TMS320F241 Data Sheet. p. 40.

La unidad CALU realiza las operaciones aritmético-lógicas, mientras que la unidad ARAU permite realizar cálculos sobre registros auxiliares para direccionamientos indirectos tanto a memoria de datos como al programa.

Finalmente la unidad de multiplicación y suma permite una rápida ejecución de operaciones, tales como algoritmo de filtros.

3.1.4. Ventajas y desventajas de los DSP's

La tecnología VLSI da la posibilidad de diseñar sistemas con la capacidad de procesamiento en tiempo real para aplicaciones en comunicaciones, control, procesamiento de imagen y multimedia.

- Los sistemas digitales son más confiables que los correspondientes sistemas análogos.

- Mayor precisión y mayor exactitud pueden ser obtenidas con sistemas digitales, comparado con los correspondientes sistemas análogos.

- Las señales digitales pueden ser almacenadas en diferentes dispositivos de almacenamiento, sin pérdida de fidelidad, y esto no es el caso para las señales análogas.

- La conversión de una señal analógica en digital obtenida muestreando la señal y cuantificando las muestras, produce una distorsión que impide la reconstrucción de la señal analógica original.

- Para muchas señales de gran ancho de banda, se requiere procesado en tiempo real.

3.1.5. Aplicaciones

Dentro del campo de las aplicaciones destacan las telecomunicaciones, la multimedia y el control de motores, esto incluye una amplia variedad de soluciones como la mejora de imágenes, el reconocimiento y la generación de la voz, la comprensión de datos para el almacenamiento y trasmisión.

A continuación se encuentran algunas aplicaciones, donde en la actualidad es indispensable contar con un DSP:

- Medicina
- Industria
- Control de motores
- Automoción
- Militar
- Telecomunicaciones
- Imagen y sonido

3.2. dsPICs

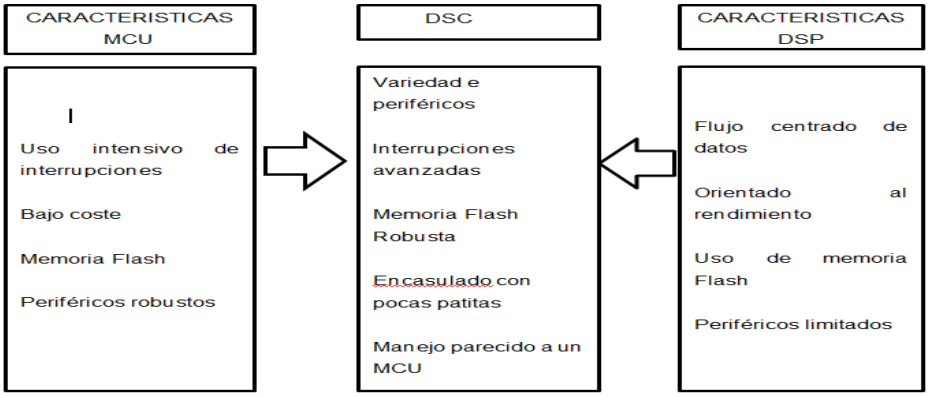
A continuación se describen las características del dispositivo a utilizar en este trabajo, pero antes se dará una introducción de que es un DSC, ya que es en esta familia donde se clasifica dicho dispositivo.

3.2.1. Controlador digital de señales (DSC)

Un DSC no es más que la combinación de la potencia de un microcontrolador de 16 bits con las prestaciones más interesantes de los DSP. Esto surge de la necesidad de responder a las modernas aplicaciones que combinan las funciones típicas del microcontrolador con las del procesamiento digital de señales.

Estos tipos de dispositivos se caracterizan por alcanzar un rendimiento de 40 MIPS e integrar memoria *flash* de alta calidad junto a novedosos recursos de hardware. Ver figura 19

Figura 19. **El DSC reúne lo mejor de un MCU Y DSP**



Fuente: elaboración propia, con programa Microft Word 2010.

Los DSC se comercializan en la actualidad, agrupados en 2 familias:

- Familia dsPIC30F
- FamiliadsPIC33F

En este trabajo se centrará en la familia dsPIC30F sin embargo se dará a conocer características generales de ambas familias, así como su diferencias.

3.2.2. Características de la familia dsPIC30F

Inicialmente la familia dsPIC30F fue fabricada por Microchip, en la tabla I se describe las características generales.

Tabla I. **Características más importantes de la familia dsPIC30F**

RECURSO	RANGO DE VALORES
Memoria del programa FLASH	12Kb-144kb
Memoria de datos RAM	512 bytes-8Kb
Memora de EEPROM	1Kb-4Kb
Temporizador de 16 bits	Hasta 5
Modulo Comparador / PWM	Hasta 8 salidas
Conversor A/D de 10 bits	500 Kbps, hasta 16 canales
UART	1-2

Fuente: elaboración propia.

Entre otras características se pueden encontrar también:

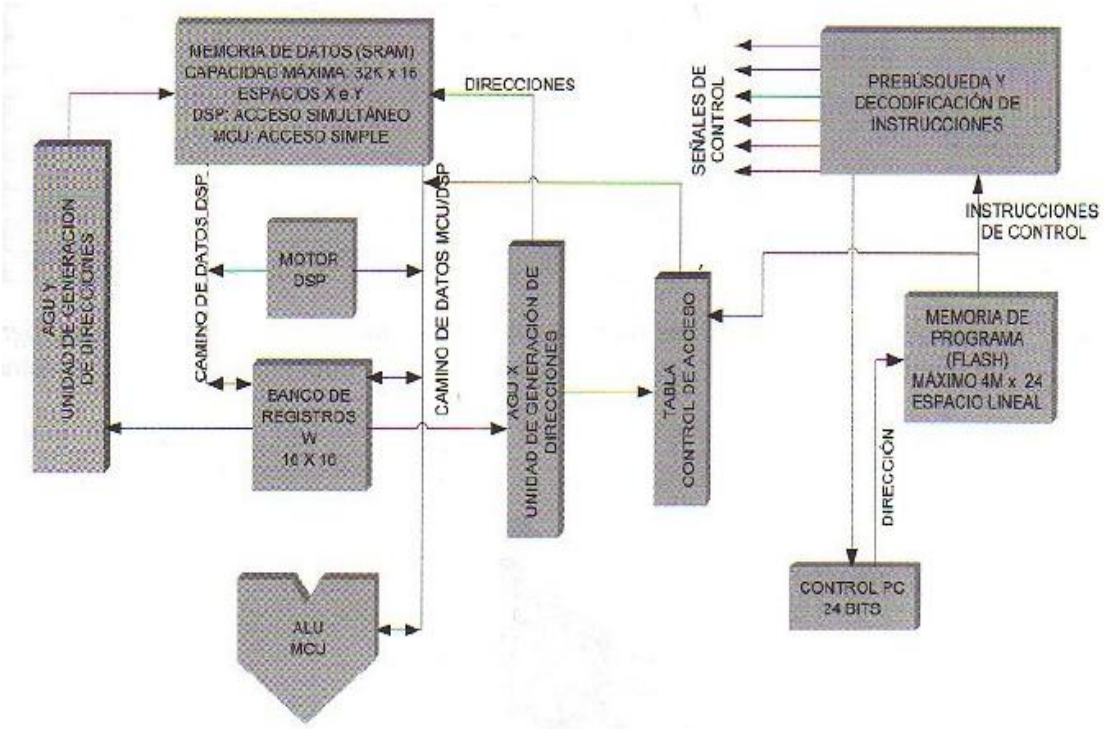
- Voltaje de alimentación admite un rango comprendido entre 2,5 y 5,5 VDC.
- Tolera una temperatura entre -40 y 85 grados Celsius y una externa entre -40 y 125 grados Celsius.
- El rendimiento alcanza los 30 MIPS cuando el voltaje tiene un valor entre 4,5 y 5,5 VDC.

En cuanto a la arquitectura de la CPU los dsPIC30F se sustentan en un núcleo RISC con arquitectura Harvard mejorada. Actuando como soporte central de la información existe un banco de 16 registros de 16 bits cada uno, se dispone de un bus de datos de 16 líneas y otro de instrucciones de 24.

Con respecto a la memoria *flash* puede llegar a alcanzar un tamaño de 4 Megabytes de instrucciones de 24 bits cada una, aunque en la actualidad solo existen modelos con capacidad máxima de 256 kilobytes.

La memoria de datos se divide en dos espacios, X e Y, que pueden ser accedidos simultáneamente en las operaciones matemáticas DSP. Toda la estructura del dsPIC30F admite operaciones MCU y DSP con un repertorio de 84 instrucciones, la mayoría de 24 bits de longitud y ejecutable en un ciclo de instrucción. Todo lo antes mencionado se puede resumir en la siguiente figura:

Figura 20. **Arquitectura básica de la CPU de los dsPIC30F**



Fuente: José Angulo. DsPIC diseño práctico de aplicaciones. P. 19.

Otra característica importante en los dsPIC30F es la de admitir hasta 45 fuentes distintas de petición de interrupción con 7 niveles de prioridad, de las cuales 5 son externas. Hay modelos de dsPIC30F que disponen de hasta 54 patitas de E/S programables.

Los dsPIC30F cuenta con una variedad de periféricos como los temporizadores, conversores AD, módulos de captura y comparación, módulos PWM para el control de motores, así como los módulos de comunicación I^2C, SPI, CAN, UART, DCI, etc.

También disponen de potentes herramientas para la gestión del sistema como perro guardia, monitor de fallo de reloj, temporizadores para la estabilización de voltaje de alimentación y frecuencia.

3.2.3. Modelos de la familia dsPIC30F

La familia dsPIC30F cuenta con 19 modelos, según la clasificación hecha por Microchip, que en la actualidad fabrica y comercializa dsPICs en 3 diferentes categorías que son:

- dsPIC30F de propósito general
- dsPIC30F para el control de sensores
- dsPIC30F para control de motores y sistemas de alimentación.

Como en este trabajo la aplicación que se le dará al dsPIC30F es para prácticas de laboratorio, solo interesa conocer el de propósito general.

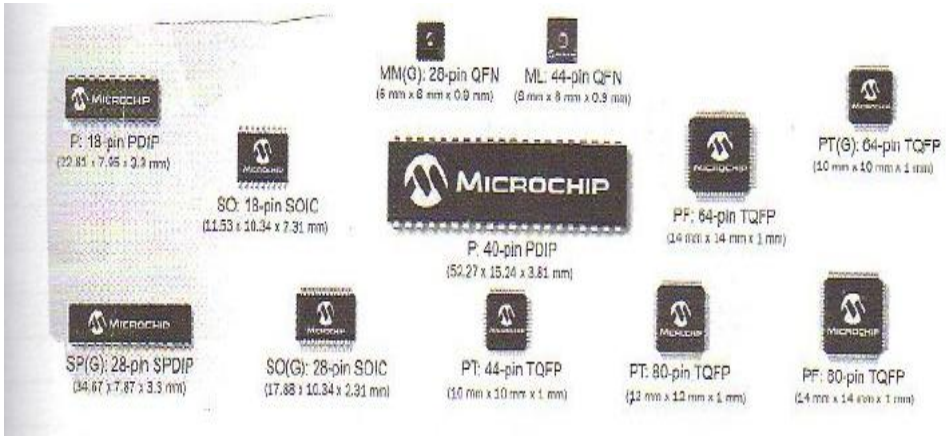
3.2.3.1. dsPIC30F de propósito general

Este grupo consta de 8 modelos diferentes orientado especialmente a las aplicaciones avanzadas de MCU de 16 bits embedidos y para el audio que precisen interfaces CODEC.

3.2.4. Encapsulados y diagrama de conexiones

Con la finalidad de soportar todo tipo de aplicaciones y diseños, los modelos están encapsulados desde 18 hasta 80 pines, unos con doble hileras de pines, tipo PDIP y SPDIP y otros están diseñados para el montaje superficial, tipo TQFT Y QFN. Estos se observan en la figura 21.

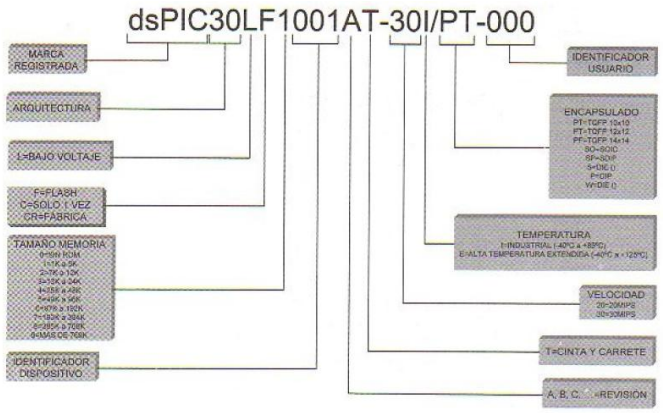
Figura 21. **Modelo de encapsulados de la familia dsPIC30F**



Fuente: José Angulo. DsPIC diseño práctico de aplicaciones. p. 25.

La nomenclatura de los dsPIC30F formada por números y letras expresan especificaciones particulares y esto se observa en la figura 22

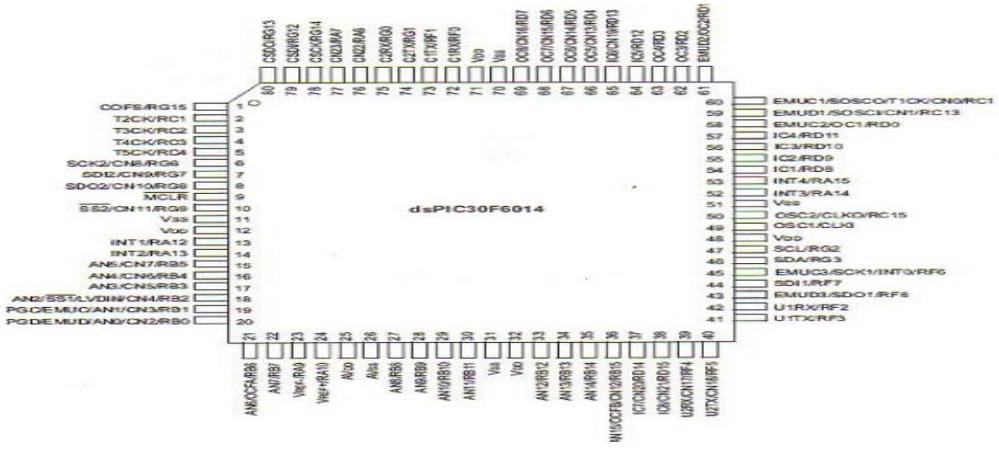
Figura 22. **Nomenclatura de la familia dsPIC30F**



Fuente: José Angulo. DsPIC diseño práctico de aplicaciones. p. 25.

El diagrama de conexiones para la familia dsPIC30F lo ha diseñado Microchip para propiciar la migración hacia modelos superiores y con más pines y esto se observa en la figura 23.

Figura 23. **Diagrama de conexiones dsPic30f6014**



Fuente: José Angulo. DsPIC diseño práctico de aplicaciones. p. 26.

3.2.5. Características de la familia dsPIC33F

De este tipo de dsPIC no será utilizado en el siguiente trabajo, pero es importante dar a conocer sus características principales, así como la diferencia con la familia dsPIC30F, ya que en un futuro basándose en el presente trabajo se puede llevar a cabo prácticas utilizando este dispositivo.

El voltaje de alimentación de la familia dsPIC33F admite un rango que va desde los 2 a los 3.6 VDC. Con respecto al rango de temperatura es idéntica a los de la familia dsPIC30F.

El rendimiento máximo alcanza los 40 MIPS cuando el voltaje de alimentación es de 3.3 VDC. Estas características se observan en la siguiente tabla.

Tabla II. **Características más importantes de la familia dsPIC33F**

RECURSO	RANGO DE VALORES
Memoria de programa FLASH	Hasta 256 KB
Memoria de datos RAM	Hasta 30KB
Memoria de datos EEPROM	No disponible
Temporizadores de 16 bits	Hasta 9
Módulo de captura	Hasta 8 entradas
Modulo comparador / PWM	Hasta 8 salidas
Conversor A/D de 10 bits	2.2 Mbps, hasta 32 canales

Fuente: elaboración propia.

3.2.6. Modelos de la familia dsPIC33F

Son 27 modelos diferentes que fabrica Microchip en la actualidad y los ha clasificado según sus aplicaciones:

- dsPIC33F de propósito general
- dsPIC33F para el control de motores y sistemas de alimentación

3.2.7. Diferencia entre las familias dsPIC30F y dsPIC33F

A pesar de ser muy similares existen diferencias como el rango de voltaje soportado por cada uno es diferente, así como su voltaje óptimo para su mejor rendimiento. También difieren en las E/S y la memoria *flash* que alcanza 144 KB en los 30 F y 256 KB en los 33 F. Finalmente los dsPIC33F cuentan con más interrupciones y con un controlador de DMA.

En la siguiente tabla se observa las diferencias más importantes.

Tabla III. **Diferencias entre las familias dsPIC30F y dsPIC33F**

dsPIC30F	dsPIC33F
26 modelos disponibles	27 modelos disponibles
5 temporizadores	9 temporizadores
Alimentación de 2 a 5.5 V	Alimentación de 2 a 3.6 V
Rendimiento: 30MIPS a 4,5 o 5,5 V	Rendimiento: 40 MIPS a 3,3 V
Memoria FLASH de 144 KB	Memoria FLASH de 256 KB
Memoria SRAM de 8 KB	Memoria SRAM de 30KB
Abundantes periféricos	Mas periféricos

Fuente: elaboración propia.

3.3. dsPIC30F4013

El microcontrolador elegido para realizar las prácticas en este trabajo, que se encuentra en capítulo 5 es el dsPIC30F4013, perteneciente a la familia de propósito general, que ha sido seleccionado por varias de sus características.

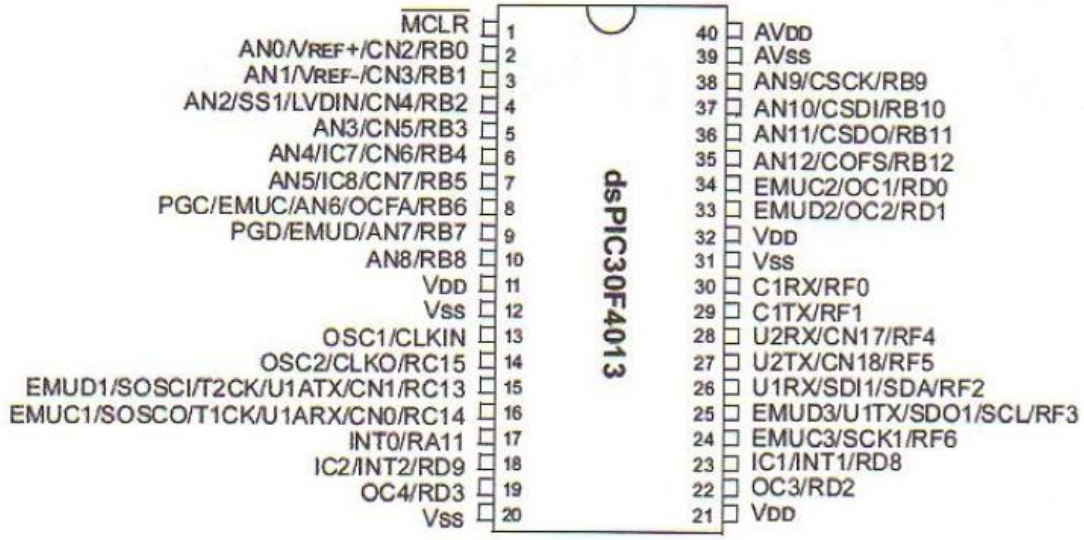
En primer lugar pertenece a una familia de propósito general, lo que lo hace interesante para poder entrar en el estudio de los dsPIC.

Por otro lado se trata de un DCS con encapsulado PDIP (Plastic Dual *In*-Line Package) lo que permite mayor sencillez para el montaje, y además porque es uno de los dsPIC permitidos por la placa entrenadora que se utilizará en cada práctica.

3.3.1. Descripción de patillas

Este dsPIC se comercializa en varios encapsulados. El seleccionado para las prácticas es el PDIP de 40 patitas, ya que es el único que admite la placa entrenadora. El diagrama de conexionado se observar en la figura 24.

Figura 24. **Diagrama de conexión del dsPIC30F4013**



Fuente: Microchip. dsPIC30F4013 data sheet. p. 2.

3.3.2. Características principales

Las características más importantes del dsPIC30F4013 son las siguientes:

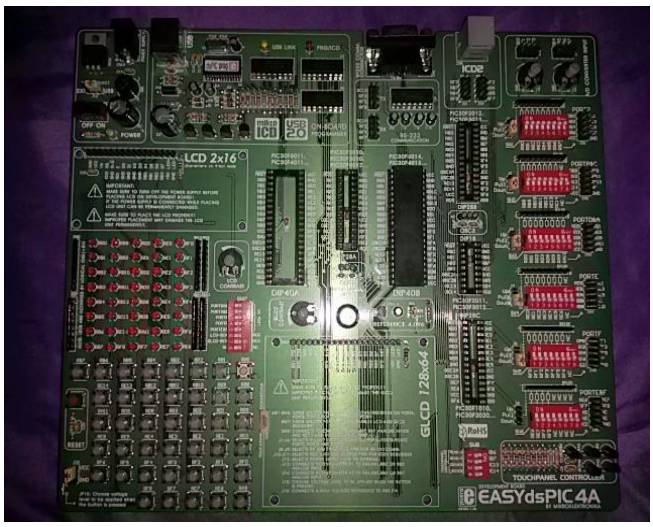
- Memoria
    - Memoria de programa de 48 kb de capacidad. Hasta 16 k instrucciones
    - 2048 bytes de memoria SRAM
    - 1024 bytes de memoria EEPROM
    - 16 registros de trabajo de 16 bits cada uno
- Periféricos
    - 5 temporizadores de 16 bits.
    - 4 módulos de captura de 16 bits.
    - 4 módulos comparadores o de salida PWM de 16 bits.
    - 2 módulos UART
    - 1 módulo SPI
    - 1 módulo CAN
    - 1 módulo I^2C
- Características analógicas
    - Conversor analógico digital de 12 bits
    - Ratio de conversión de 100 ksps
    - Hasta 13 canales de entrada

3.4. Placa entrenadora EasydsPIC4A

Ahora se describen las características principales de la herramienta principal que hizo posible este trabajo, esta es una tarjeta muy completa de la empresa Mikroelectronica, en la cual ya viene integrada una serie de periféricos que facilitan la utilización en las prácticas que se realizaran en el capítulo 5 y 6.

A continuación se presenta una imagen real de la placa utilizada que fue prestada del Laboratorio de Electrónica de la Facultad de Ingeniería. Ver figura 25.

Figura 25. **Foto Real de la placa entrenadora EasydsPIC4A**

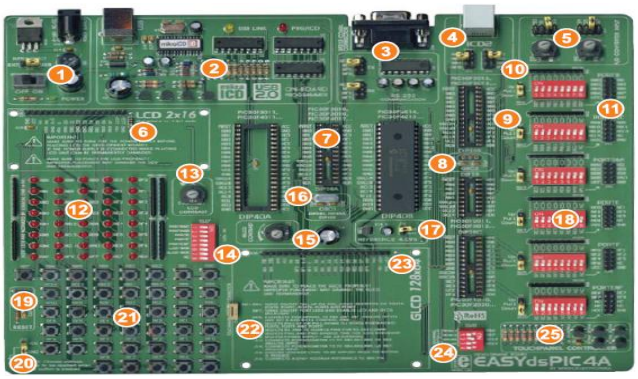


Fuente: Laboratorio de Electrónica, T-1, Facultad de Ingeniería.

3.4.1. Características principales

En la figura 26 se describen las características con las que cuenta la tarjeta entrenadora EasydsPIC4A.

Figura 26. **Placa entrenadora EasyPIC4A**



1. Fuente de alimentación externa de 8v a 16v AC/DC
2. programador con mikroICD USB 2.0
3. RS-232
4. Programador externo ICD2
5. Potenciómetros de entrada para prueba de A/D
6. Pantalla LCD 2x16
7. *Soket* para dsPIC DIP18,DIP28 y DIP40.
8. Conector para oscilador
9. Jumpers para determinada entrada
10. Red de resistencias 8x10k.
11. Puerto para conectores de acceso directo.
12. Pin I/O correspondiente a cada LED
13. Potenciometro para contraste de LCD
14. *Switch* para todos los leds en los puertos A,B,C,D, y F.
15. Poteciometro para contraste de la GLCD.
16. OSC1 10Mhz
17. Votaje de referencia 4.096V.
18. Grupos de Switchs que habilitan *pull-up/pull-down*
19. Botón de *reset*.
20. Jumper J15 para seleccionar estado alto o bajo de los pines.
21. 41 *push-buttons* para controlar los pines del microcontrolador
22. CN11 conector del panel touch.
23. Conector para la GLCD
24. *Switch* para habilitar/deshabilitar entre el panel touch y el microcontrolador.
25. Controlador del panel *touch*.

Fuente: http://www.mikroe.com/downloads/get/333/easydspic4a_manual_v100.

Consulta: 17 de julio de 2013.

A continuación se describen algunas características que se consideran importantes, ya que fueron las utilizadas para desarrollar las prácticas de este trabajo.

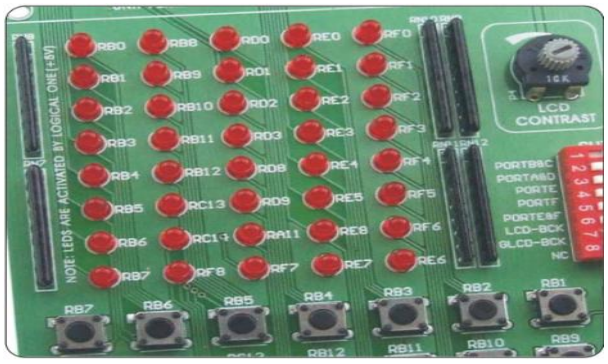
3.4.2. Configuraciones y diagramas principales

A continuación se describen los periféricos de la tarjeta entrenadora EasydsPIC4A, que fueron utilizados en este trabajo, así como sus diagramas.

3.4.2.1. LEDs

Diodos emisores de luz son componentes comúnmente usados para mostrar el estado digital de un pin. La EASYdsPICA cuenta con 40 LEDs conectados a los puertos A, B, C, D, E y F del microcontrolador. Ver figura 27.

Figura 27. **LEDs placa Entrenadora EasyPIC4A**

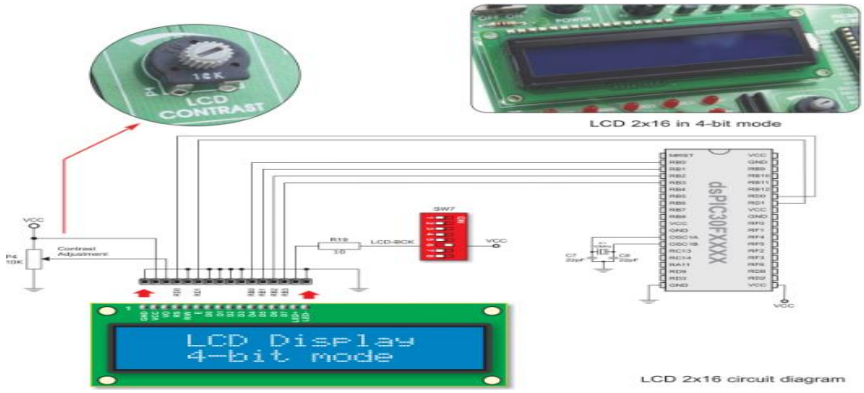


Fuente: http://www.mikroe.com/downloads/get/333/easydspic4a_manual_v100.
Consulta: 17 de julio de 2013.

3.4.2.2. LCD 2x16

LCD estándar la cual puede ser usada para desplegar mensajes en dos líneas y en cada línea puede contener 16 caracteres alfanuméricos. La LCD se comunica con el micro controlador por medio de un bus de 4 bits. El diagrama se muestra en la figura 28.

Figura 28. **Diagrama de conexión de la LCD 2x16**

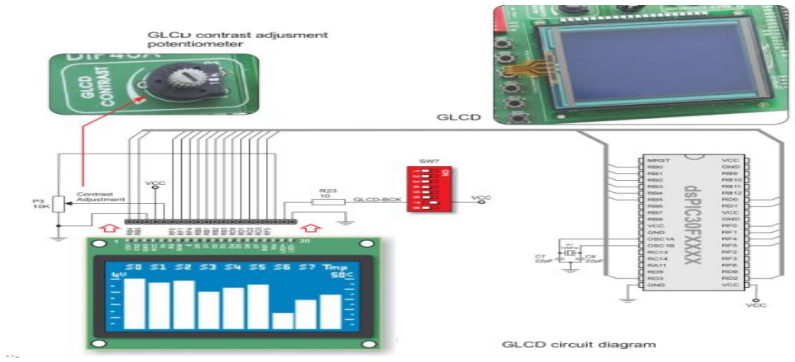


Fuente: http://www.mikroe.com/downloads/get/333/easydspic4a_manual_v100.
Consulta: 17 de julio de 2013.

3.4.2.3. GLCD

Una LCD gráfica provee de una forma avanzada de mostrar texto y gráficas al mismo tiempo. La GLCD cuenta con una pantalla de 128 x 64 pixeles, lo cual es suficiente para mostrar mensajes en forma de dibujos o gráficas. El diagrama se muestra en la figura 29.

Figura 29. **Diagrama de conexión de la GLCD**



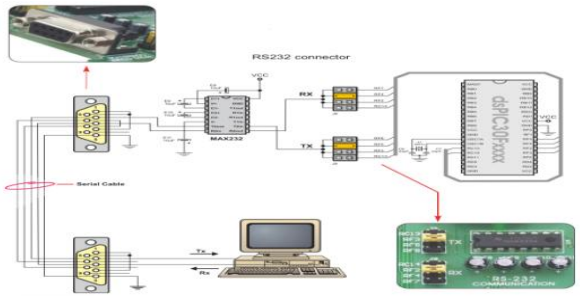
Fuente: http://www.mikroe.com/downloads/get/333/easydspic4a_manual_v100.
Consulta: 17 de julio 2013.

3.4.2.4. Comunicación RS-232

Comunicación RS-232 es un modo de trasferencia de datos punto a punto. Es comúnmente usada en las aplicaciones para la adquisión de datos entre el microcontrolador y la computadora.

Dado que los niveles de voltaje entre el micro controlador y la pc no son los mismos, se debe utilizar un MAX232, ver la figura 30 de como conectar el circuito para utilizar este periférico.

Figura 30. **Diagrama de conexión RS-232**

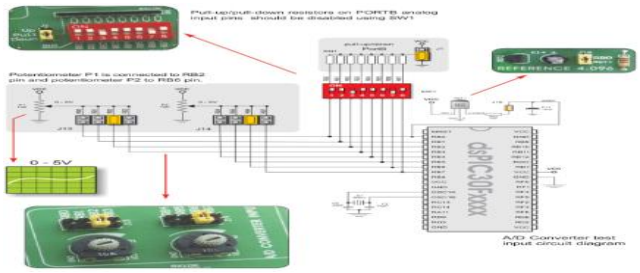


Fuente: http://www.mikroe.com/downloads/get/333/easydspic4a_manual_v100.
Consulta: 17 de julio de 2013.

3.4.2.5. Entrada de prueba para el convertidor A/D

La placa entrenadora tiene una tarjeta con dos potenciómetros para demostrar la operación del convertidor análogo a digital (ADC), ambos potenciómetros cuentan con una salida de 0 a 5 voltios. Estas señales analógicas pueden ser llevadas a dos pines de entrada analógica diferentes al mismo tiempo. Ver la configuración en la figura 31.

Figura 31. **Diagrama de conexión A/D**



Fuente: http://www.mikroe.com/downloads/get/333/easydspic4a_manual_v100.
Consulta: 17 de julio de 2013.

4. DESARROLLO DE PRÁCTICAS DE LABORATORIO CON MATLAB

En este capítulo se realizarán 3 prácticas utilizando los comandos de MATLAB vistos anteriormente (ver capítulo 2). Con el desarrollo de estas prácticas el estudiante tendrá una visión más amplia de todas las aplicaciones que se pueden realizar en el tratamiento de señales digitales, utilizando este software.

4.1. Generando y graficando señales continuas y discretas

En esta práctica se describen todos los comandos necesarios para generar y graficar señales en tiempo continuo y discreto, se realizará paso a paso como utilizar estos comandos.

4.1.1. Objetivos

Generar diferentes tipos de señales como lo son sinusoidales, cuadradas, dientes de sierra y escalón, y realizar la gráfica en tiempo continuo y discreto.

Describir los comandos para poder generar y graficar las señales continuas y discretas.

4.1.2. Procedimiento

Se realizaron diferentes ejemplos con cada tipo de señales mostrando el código de cómo se realizó y el resultado, en este caso sería la gráfica correspondiente de cada señal, la práctica está dividida en 2 partes que son las siguientes:

4.1.3. Señales continuas

Como ya se ha visto en capítulos anteriores, existen varios tipos de señales continuas empezando con las exponenciales.

4.1.3.1. Señales exponenciales

Lo primero que se hizo es generar un archivo nuevo .m luego se escribió cada instrucción descrita en la figura 32. Luego se ejecutó el código para observar el resultado.

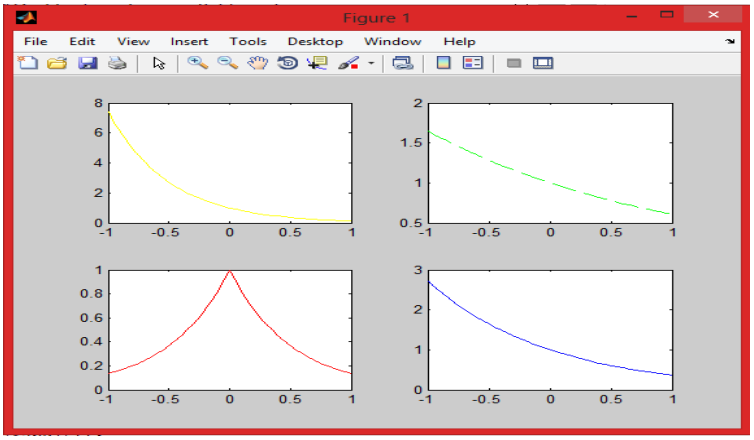
Figura 32. **Código de la práctica de señales exponenciales**

```
% practica genrando señales continuas exponenciales

T=0.05; % vector que representa la secuencia temporal

t=[-1:T:1]; % representa la separacion temporal( en segundos)

%Ahora generamos las siguientes señales

x=exp(-t);
x1=exp(-2*t);
x2=exp(-t/2);
x3=exp(-2*abs(t));

%Las graficamos en una matrix grafica de 2x2.

subplot(2,2,1);plot(t,x1,'-y');
subplot(2,2,2);plot(t,x2,'--g');
subplot(2,2,3);plot(t,x3,'r');
subplot(2,2,4);plot(t,x,'-b');
```

Fuente: elaboración propia, con programa MATLAB R2010b.

Aquí se pueden observar claramente los comandos utilizados para esta práctica, nuevamente se ejecutó el código y se obtuvo el siguiente resultado, ver figura 33.

Figura 33. **Señales exponenciales**

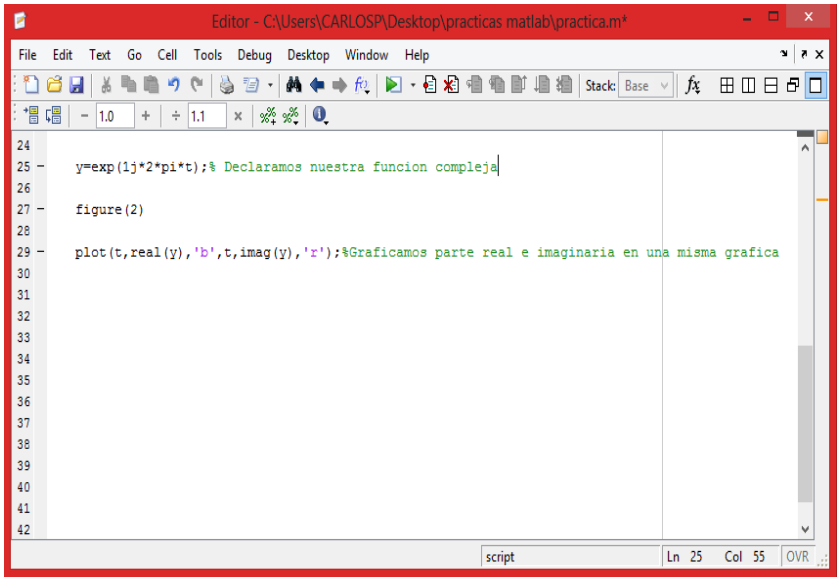


Fuente: elaboración propia, con programa MATLAB R2010b.

Con este resultado se pudo observar de una manera fácil y sencilla como generar este tipo de señales, se realizó otro ejemplo de este mismo tipo de señales, con una señal exponencial compleja.

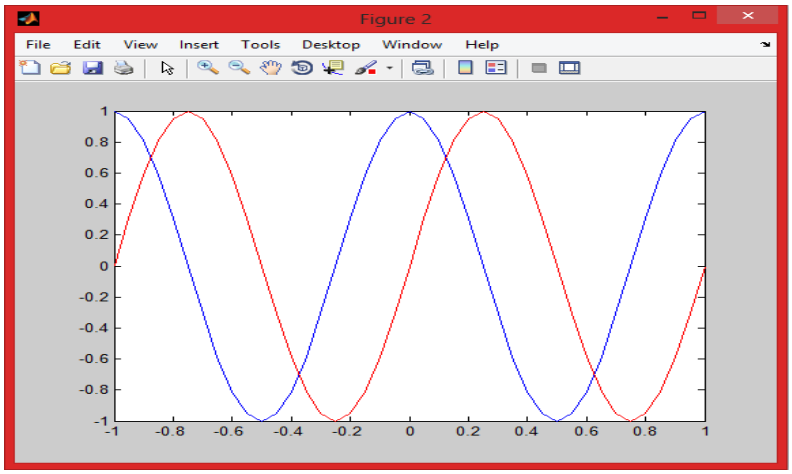
Para generar una señal compleja por ejemplo $y(t) = e^{j2\pi t}$, se realizó de la siguiente manera, ver figuras 34 y 35.

Figura 34. **Código fuente de una señal compleja**



Fuente: elaboración propia, con programa MATLAB R2010b.

Figura 35. **Señal compleja**



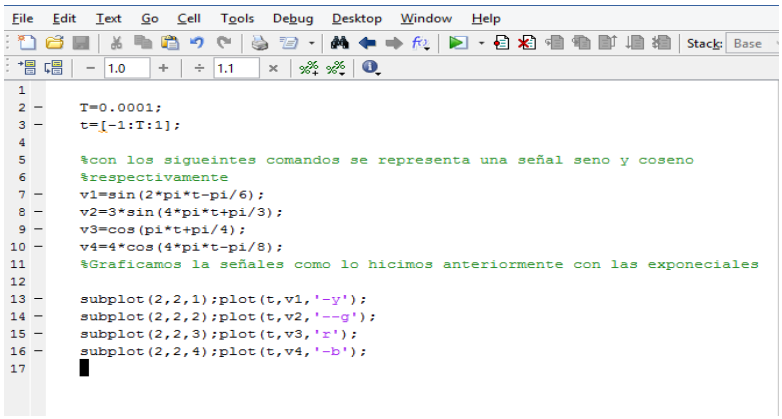
Fuente: elaboración propia, con programa MATLAB R2010b.

En la figura 35 se observa el resultado gráfico de una señal exponencial compleja, de la matemática se sabe que por el teorema de Euler este tipo de señales se puede descomponer en seno y coseno, y esto se puede observar en la gráfica anterior.

4.1.3.2. Señales sinusoidales

Las sinusoides reales son generadas de manera directa por MATLAB, ahora se describe cómo se generan 4 señales: 2 seno y 2 coseno manipulando sus propiedades.

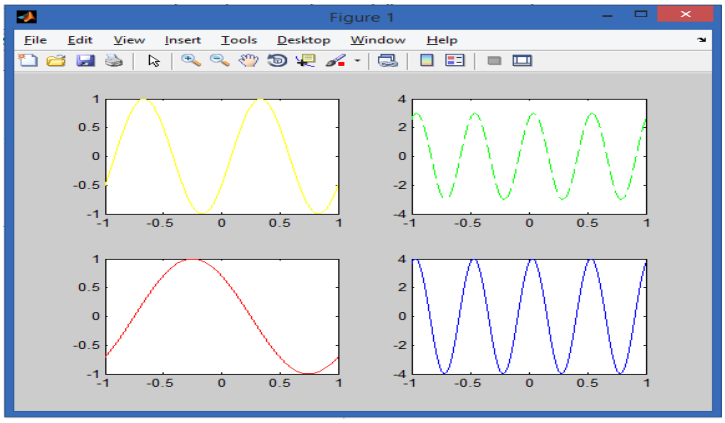
Figura 36. **Código fuente de señales sinusoidales**



Fuente: elaboración propia, con programa MATLAB R2010b.

Como se observa en la figura 36 los comandos son simples, solo es de tener el conocimiento para poder manipularlos sin caer en errores, en la figura 37 se observa el resultado.

Figura 37. **Señales sinusoidales**

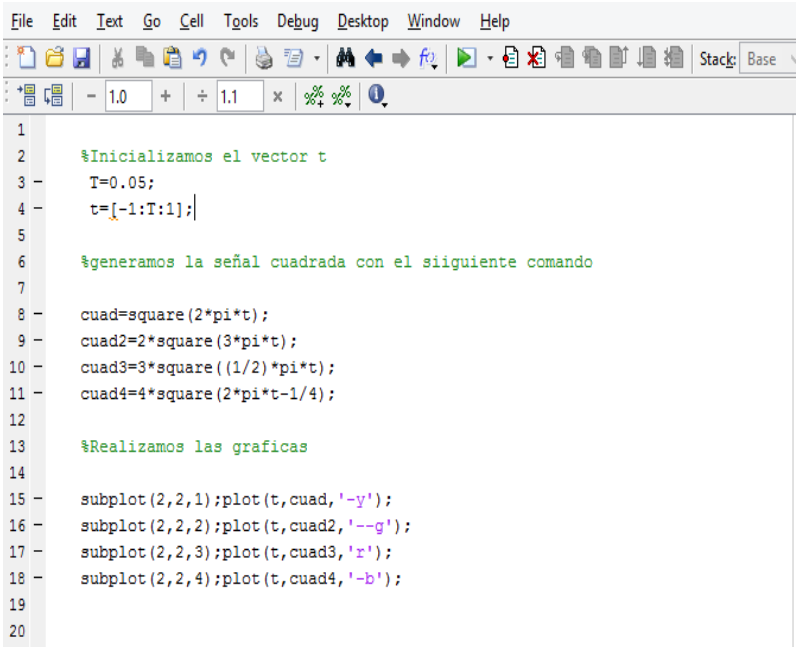


Fuente: elaboración propia, con programa MATLAB R2010b.

4.1.3.3. Señales cuadradas

A continuación se describe cómo generar una señal cuadrada usando el siguiente código. Ver figura 38.

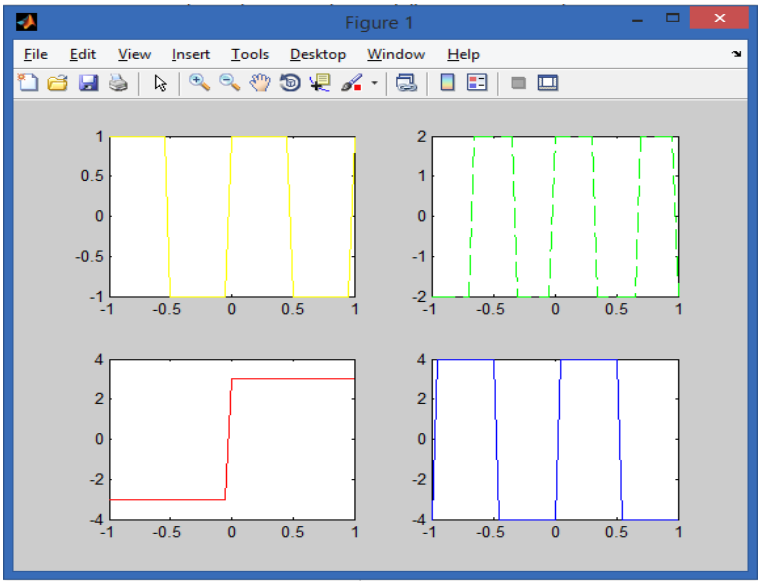
Figura 38. **Código fuente de señales cuadradas**



Fuente: elaboración propia, con programa MATLAB R2010b.

Como se observa en el código escrito, son 4 señales cuadradas diferentes. Ver figura 39.

Figura 39. **Señales cuadradas**



Fuente: elaboración propia, con programa MATLAB R2010b.

En la figura anterior se observa, que las pendientes no son infinitas, esto ocurre porque el número de puntos es bajo, esto se puede mejorar definiendo otro vector de tiempo y volviendo a graficar.

4.1.3.4. Señales diente de sierra

A continuación se genera una señal diente de sierra periódica con el siguiente código, ver figura 40.

Figura 40. **Código fuente de señales diente de sierra**

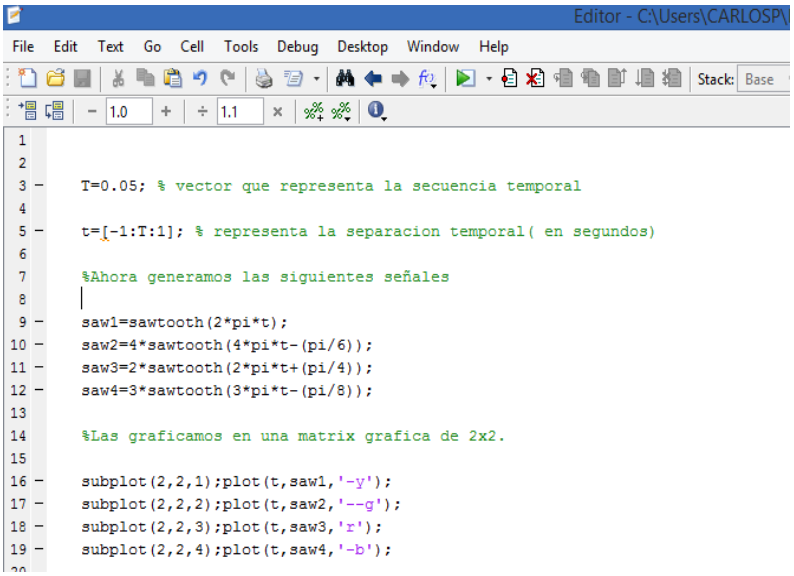
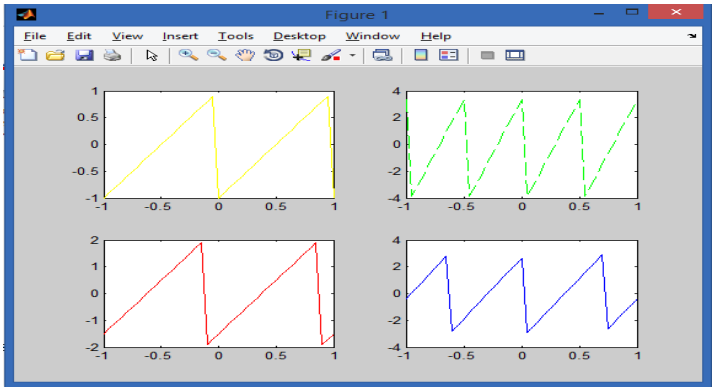


Figura: elaboración propia, con programa MATLAB R2010b.

Se ejecutan los comandos y se obtienen las siguientes gráficas. Ver figura 41.

Figura 41. **Señales diente de sierra**

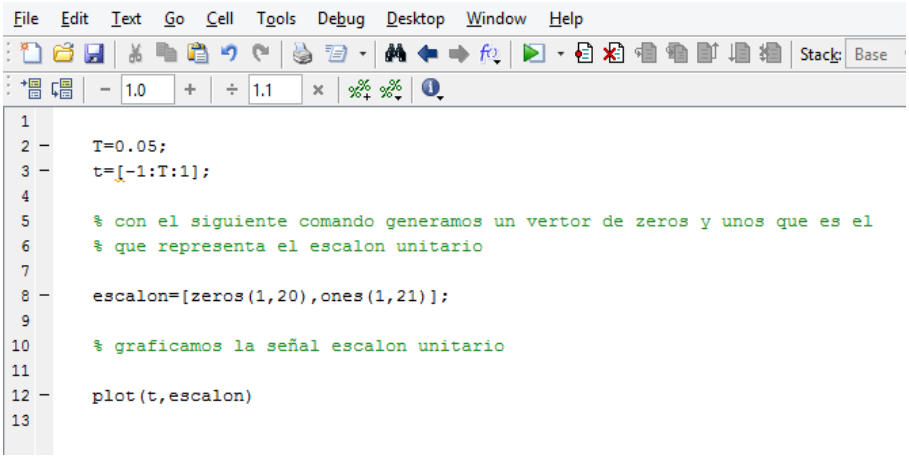


Fuente: elaboración propia, con programa MATLAB R2010b.

4.1.3.5. Señal escalón unitario

Terminado con las señales continuas, se realiza la última gráfica, la cual es de un escalón unitario, los comandos son los siguientes, ver figura 42.

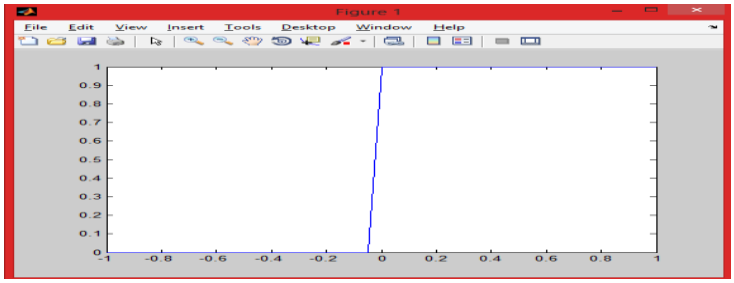
Figura 42. **Código fuente escalón unitario**



Fuente: elaboración propia, con programa MATLAB R2010b.

Se ejecuta los comandos y se obtiene la gráfica a dicha señal. Ver figura 43.

Figura 43. **Escalón unitario**



Fuente: elaboración propia, con programa MATLAB R2010b.

Con esta gráfica se termina con esta parte de la práctica, a continuación se describe cómo se genera y como se grafican las señales discretas.

4.1.4. Señales discretas

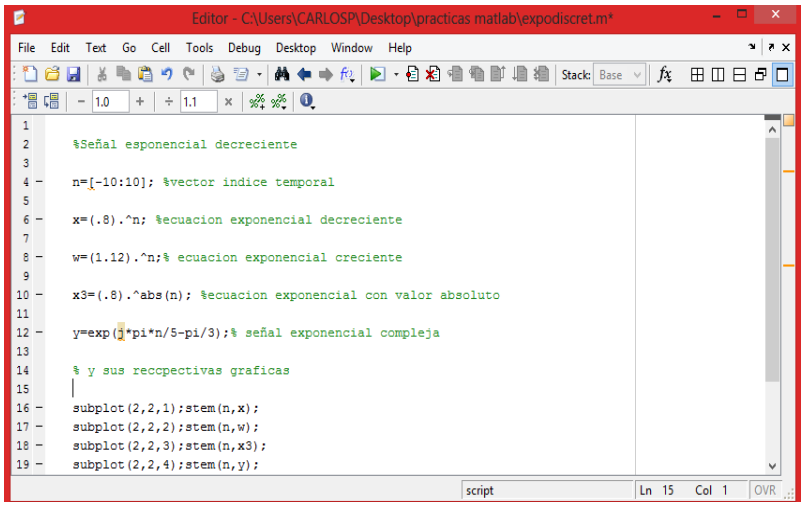
En esta segunda parte de la práctica, se describen los comandos necesarios para generar señales discretas, así como también el comando para graficar dichas señales, ya que difiere al de señales continuas.

Como primer paso se procede por abrir un nuevo archivo .m y siempre se debe recordar utilizar el comando *clear* para borrar cualquier valor que pudo haber quedado en la memoria de MATLAB.

4.1.4.1. Señal exponencial discreta

Para generar una señal discreta $x[n]$, se debe primero definir un vector índice temporal, normalmente denotado por la letra n. Se generan las siguientes señales exponenciales decrecientes, crecientes, valor absoluto y complejo. Ver código en figura 44.

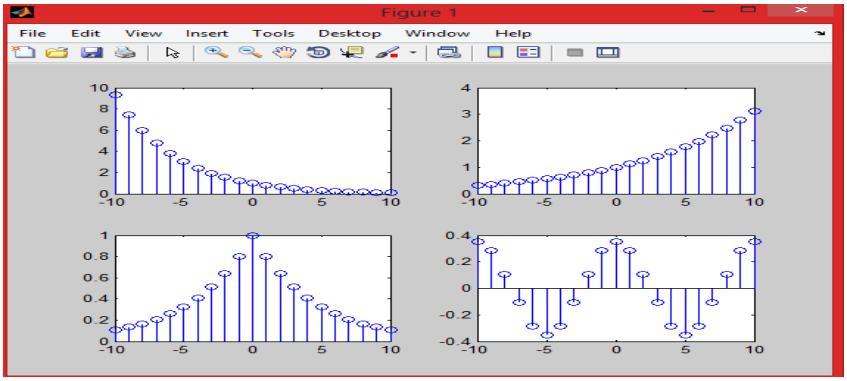
Figura 44. **Código fuente señales discretas exponenciales**



Fuente: elaboración propia, con programa MATLAB R2010b.

Se ejecuta el código como se realizó en las demás prácticas y se observa la gráfica en tiempo discreto de estas señales. Ver figura 45.

Figura 45. **Señales discreta exponenciales**

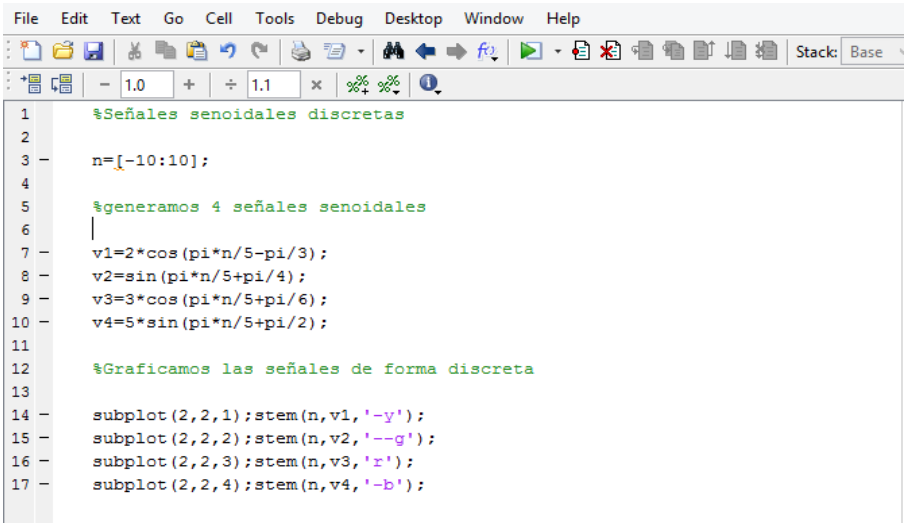


Fuente: elaboración propia, con programa MATLAB R2010b.

4.1.4.2. Señales sinusoidales discretas

Se generan las siguientes señales sinusoidales, como se realizó en señales continuas, con fase, amplitud y frecuencia angular, solo que ahora se grafica con el nuevo comando aprendido en la práctica anterior, ver figura 46.

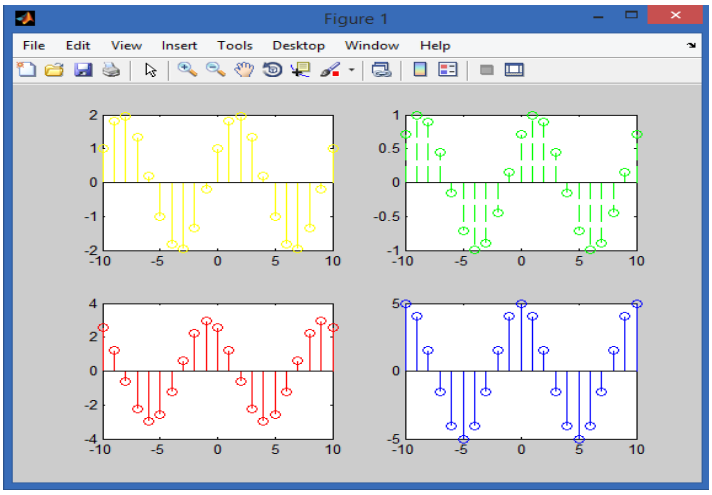
Figura 46. **Código fuente señales sinusoidales discretas**



Fuente: elaboración propia, con programa MATLAB R2010b.

Se observa el código muy similar a la forma de trabajar estas señales a las que se vió en tiempo continuo, con la salvedad que a la hora de graficar es donde cambia el comando, se ejecuta el código y se obtiene lo siguiente. Ver figura 47.

Figura 47. **Señales sinusoidales discretas**

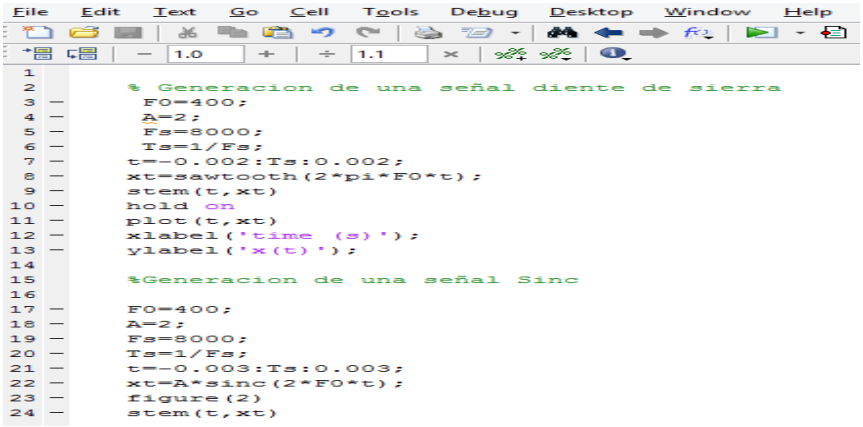


Fuente: elaboración propia, con programa MATLAB R2010b.

4.1.4.3. Otras señales en tiempo discreto

A continuación se describen 2 señales más, en esta parte de la práctica se incluyen: 2 diferentes tipos de señal, pero de igual importancia para el tratamiento de señales digitales que es el tema principal de este trabajo. Ver las siguientes señales en la figura 48.

Figura 48. **Código fuente de otras señales en tiempo discreto**

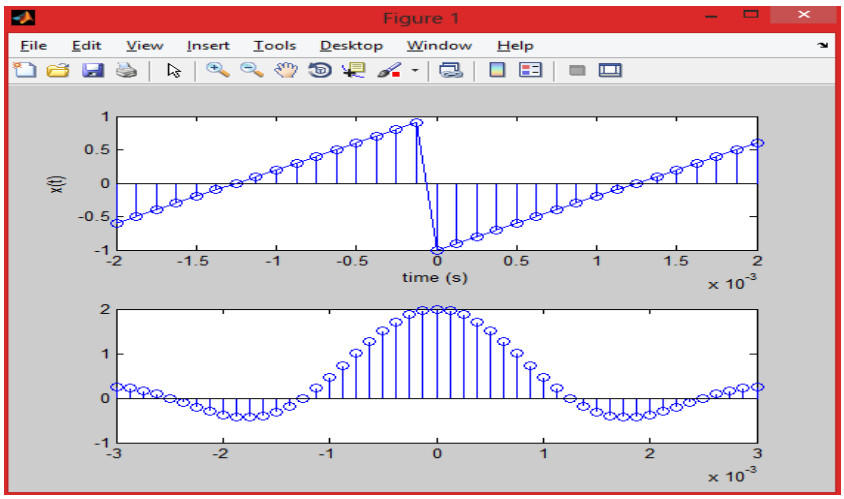


```
% Generacion de una señal diente de sierra
 F0=400;
 A=2;
 Fs=8000;
 Ts=1/Fs;
t=-0.002:Ts:0.002;
xt=sawtooth(2*pi*F0*t);
stem(t,xt)
hold on
plot(t,xt)
xlabel('time (s)');
ylabel('x(t)');

%Generacion de una señal Sinc

F0=400;
A=2;
Fs=8000;
Ts=1/Fs;
t=-0.003:Ts:0.003;
xt=A*sinc(2*F0*t);
figure(2)
stem(t,xt)
```

Fuente: elaboración propia, con programa MATLAB R2010b.

Como se puede observar en el código, las señales son diente de sierra y *Sinc* y sus respectivas gráficas se pueden ver en la figura 49.

Figura 49. **Otras señales en tiempo discreto**



Fuente: elaboración propia, con programa MATLAB R2010b.

Con estas gráficas se termina esta práctica, donde se pudo observar varios comandos que son de mucha utilidad y nada complicados de usar.

4.1.5. Recomendaciones

Realizar la misma práctica variando diferentes parámetros como lo son: amplitud, fase, frecuencia angular, frecuencia de muestro entre otros.

Investigar más señales diferentes de las vistas acá, y ver como se grafican. Aquí se describieron las más conocidas, pero en el campo del tratamiento de señales digitales existen muchas más.

4.2. Diseño de un filtro IIR

En esta práctica se describe como diseñar filtros IIR en MATLAB de una manera simple, pero siempre teniendo cuidado de utilizar bien los comandos. A continuación se realizará el diseño de 2 filtros pasabanda utilizando diferentes métodos como los son *butterworth* y *elíptico*, se hará una comparación de estos filtros, luego se utilizará uno para filtrar una señal de audio.

4.2.1. Objetivos

Diseñar 2 filtros IIR de diferente tipo, para así poder observar y conocer las diferencias entre ellos.

Aprender a utilizar la herramienta de MATLAB llamada FVTOOL, ya que es de mucha utilidad a la hora del estudio y diseño de filtros digitales.

4.2.2. Procedimiento

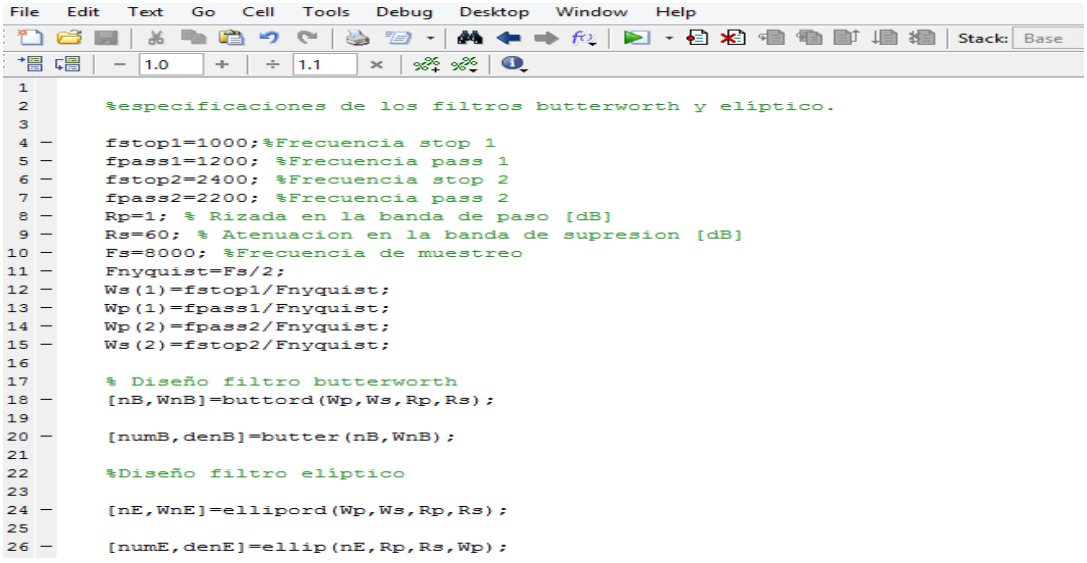
Paso 1: leer la teoría de filtros digitales en MATLAB y todo lo respecto a comandos de procesamiento de audio en MATLAB, que se encuentra en este trabajo en el capítulo 3, para así tener el conocimiento de los comandos que utilizaran en esta práctica.

Paso 2: los filtros diseñados son de tipo IIR con 2 métodos diferentes ya mencionados anteriormente, que para ambos se utilizaran las mismas especificaciones, que son las siguientes:

- Frecuencia stop 1 1000hz
- Frecuencia stop 2 2400hz
- Frecuencia de pass 1 1200hz
- Frecuencia de pass 2 2200hz
- Rizado en banda de paso 1dB
- Atenuación de banda de supresión 60 dB
- Frecuencia de muestreo 8000hz

Paso 3: con las especificaciones anteriormente dadas, se va a escribir el código en el editor de MATLAB y luego se ejecuta el código, ver las figuras 50 y 51.

Figura 50. **Código fuente filtro IIR parte 1**



```
%especificaciones de los filtros butterworth y elíptico.

fstop1=1000;%Frecuencia stop 1
fpass1=1200; %Frecuencia pass 1
fstop2=2400; %Frecuencia stop 2
fpass2=2200; %Frecuencia pass 2
Rp=1; % Rizada en la banda de paso [dB]
Rs=60; % Atenuacion en la banda de supresion [dB]
Fs=8000; %Frecuencia de muestreo
Fnyquist=Fs/2;
Ws(1)=fstop1/Fnyquist;
Wp(1)=fpass1/Fnyquist;
Wp(2)=fpass2/Fnyquist;
Ws(2)=fstop2/Fnyquist;

% Diseño filtro butterworth
[nB,WnB]=buttord(Wp,Ws,Rp,Rs);

[numB,denB]=butter(nB,WnB);

%Diseño filtro elíptico

[nE,WnE]=ellipord(Wp,Ws,Rp,Rs);

[numE,denE]=ellip(nE,Rp,Rs,Wp);
```

Fuente: elaboración propia, con programa MATLAB R2010b.

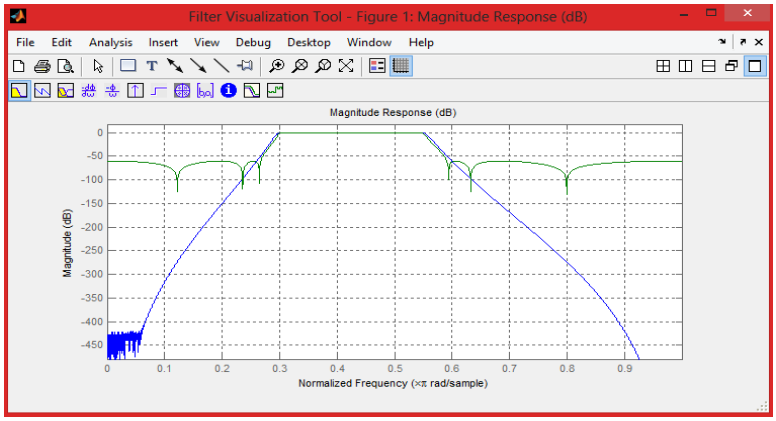
Figura 51. **Código fuente filtro IIR parte 2**

```
%Utilizamos fvtool para comparar los 2 filtros.

fvtool(numB,denB,numE,denE)

%Utilizamos el filtro pasabanda eliptico y lo aplicamos a una señal de
%audio

[xt,fs]=wavread('x1'); % Leemos la señal de audio
yt=filter(numE,denE,xt);% aplicamos el filtro eliptico
soundsc(xt,fs)%Reproducimos la eñal normal
soundsc(yt,fs)%Reproducimos la señal filtrada


%Gráficas:

subplot(2,1,1)
stem(xt)
title('Señal Original')

subplot(2,1,2)
stem(yt)
title('Señal de Salida')
```

Fuente: elaboración propia, con programa MATLAB R2010b.

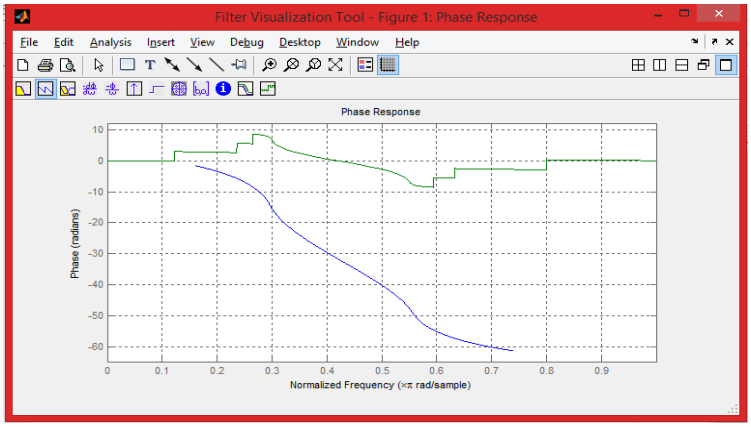
En las figuras se observa paso a paso como se crean los filtros, como se comparan y luego como se aplica una señal de audio en tiempo real, ahora se observan los resultados.

Figura 52. **Respuesta en magnitud filtro IIR**



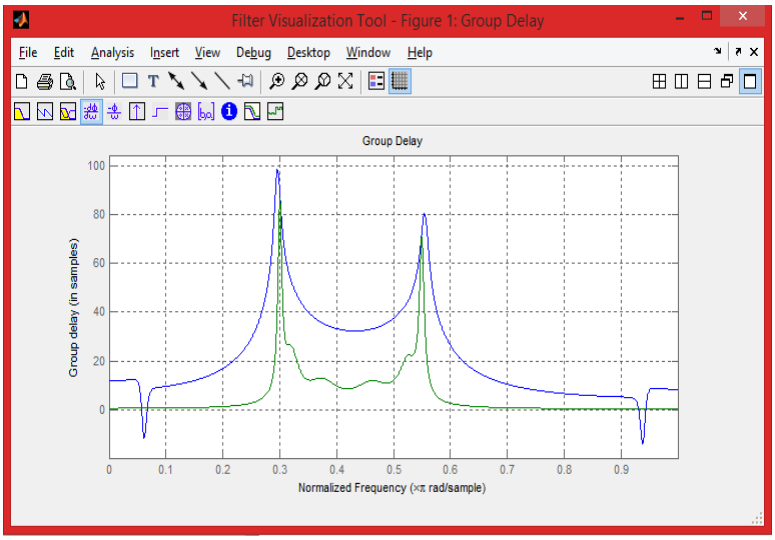
Fuente: elaboración propia, con programa MATLAB R2010b.

Figura 53. **Respuesta de fase filtro IIR**



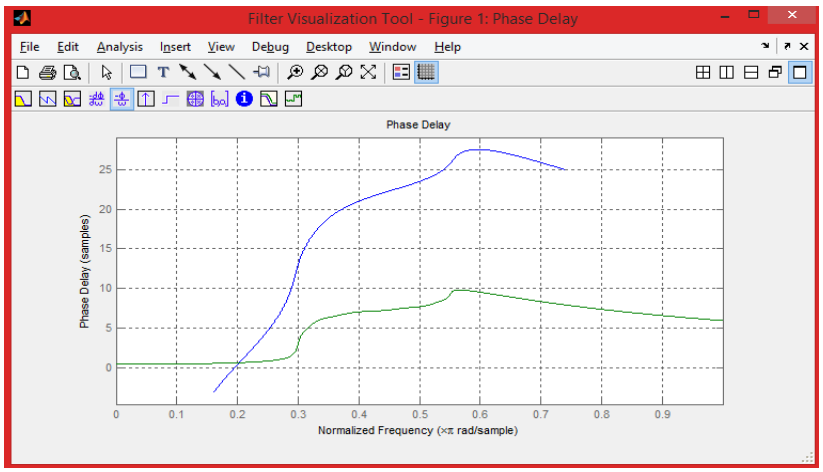
Fuente: elaboración propia, con programa MATLAB R2010b.

Figura 54. **Grupo delay filtro IIR**



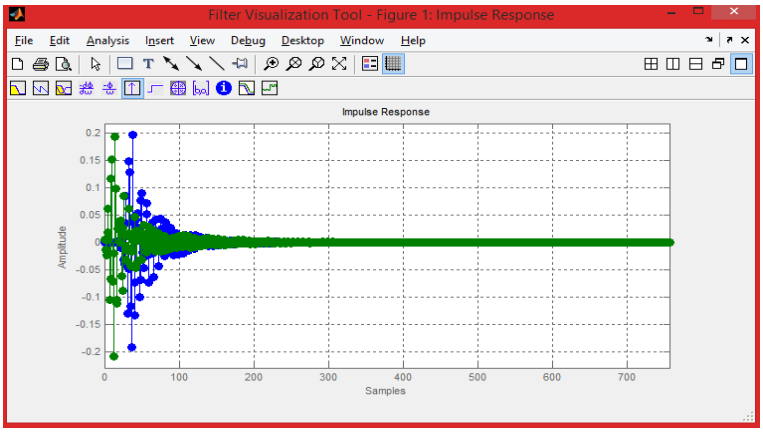
Fuente: elaboración propia, con programa MATLAB R2010b.

Figura 55. **Phase delay filtro IIR**



Fuente: elaboración propia, con programa MATLAB R2010b.

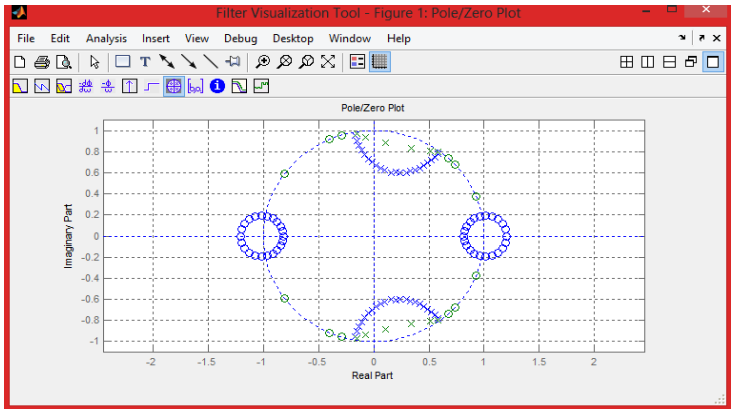
Figura 56. **Respuesta al impulso filtro IIR**



Fuente: elaboración propia, con programa MATLAB R2010b.

En las figuras anteriores se observa todo el análisis que brinda la herramienta FDTOOL, el filtro *butterworth* y el filtro *elíptico*, y según el análisis y conocimiento ya adquirido en filtros, se puede deducir que el filtro elíptico tiene una respuesta más eficiente.

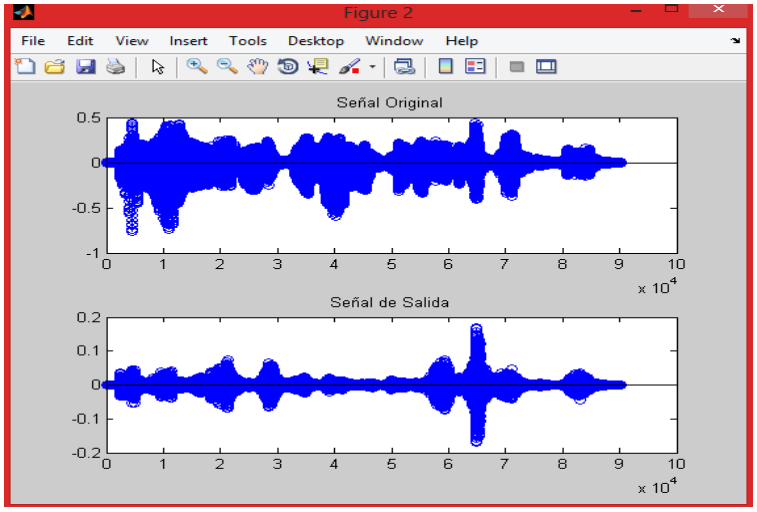
Figura 57. **Polos y ceros filtro IIR**



Fuente: elaboración propia, con programa MATLAB R2010b.

En la figura 57 se observa el diagrama de polos y zeros de los 2 filtros, de lo cual se puede concluir que ambos son filtros muy estables.

Figura 58. **Aplicando un filtro IIR a una señal de audio**



Fuente: elaboración propia, con programa MATLAB R2010b.

En la figura anterior se observa la señal de audio antes y después de ser filtrada.

4.2.3. Recomendaciones

Diseñar filtros utilizando los otros métodos como lo son Chebyshev 1 y 2, para poder observar más diferencias entre cada método y así poder realizar un buen diseño de filtros tipo IIR.

Cambiar parámetros para poder tener un escenario más amplio con respecto a diferencias entre cada uno de los filtros.

4.3. Diseño de un filtro FIR

En la última práctica sobre MATLAB, se describe el diseño de un filtro pasa banda de tipo FIR mediante el uso de ventana *Blackman*, hay muchos más tipos de ventanas con diferentes características, las cuales se puede profundizar en la teoría de filtros digitales que se encuentra en este trabajo en el capítulo 3.

4.3.1. Objetivos

Diseñar un filtro FIR en MATLAB y poder compararlo con el filtro IIR anteriormente diseñado

Conocer los comandos apropiados para el diseño de filtros FIIR en MATLAB así como la herramienta *fvtool*.

4.3.2. Procedimiento

Paso 1:leer toda la teoría respecto a filtros digitales que se encentra en el capítulo 3 de este trabajo, para así tener base para la realización de esta práctica.

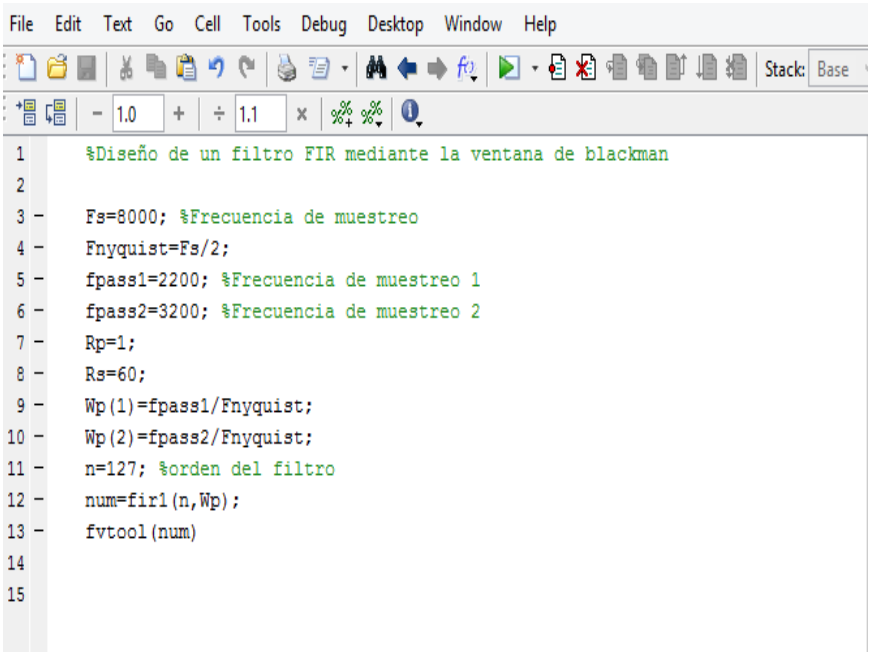
Pasó 2: se diseña un filtro FIR pasa banda mediante la ventana de Blackman, las características de dicho filtro son las siguientes:

- Fs = 8 000 Hz
- Fpass1 = 2 200 Hz
- Rp = 1 dB
- Rs = 60 dB

- Orden = 127

Paso 3: con las especificaciones anteriormente dadas, se procede a escribir el código en el editor de MATLAB y seguidamente se ejecuta el código, ver figura 59.

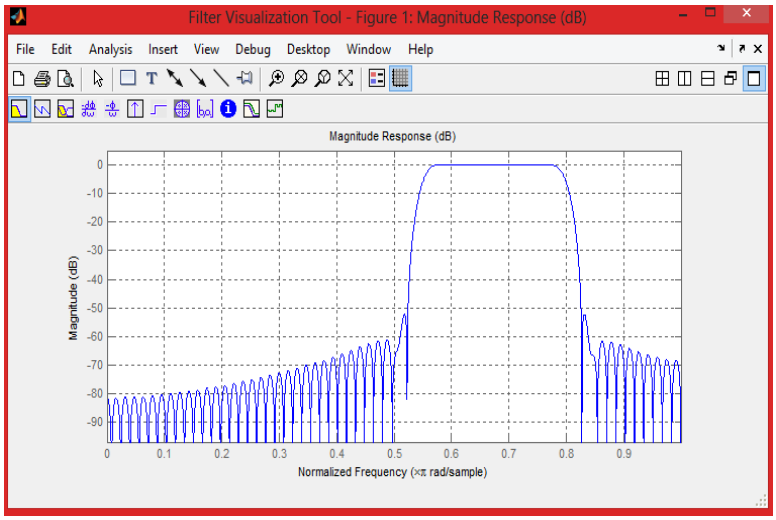
Figura 59. **Código fuente filtro FIR**



Fuente: elaboración propia, con programa MATLAB R2010b.

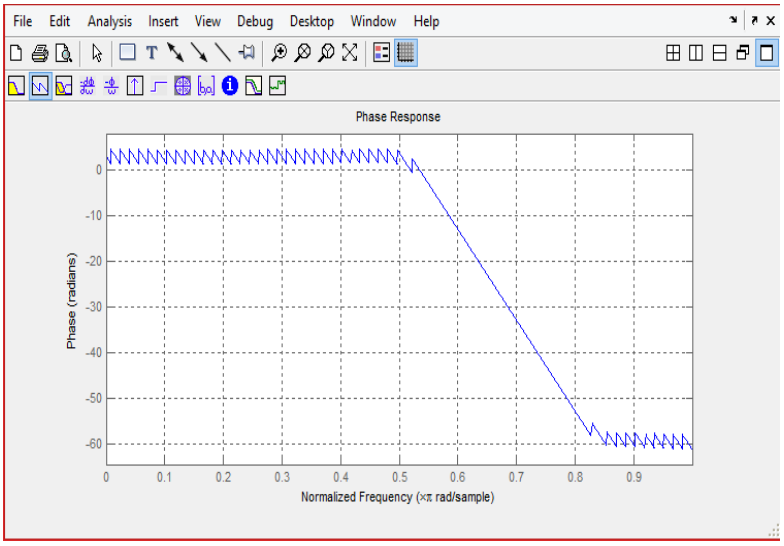
Se observa la respuesta en frecuencia del filtro previamente diseñado. Ver figura 60.

Figura 60. **Respuesta de magnitud filtro FIR**



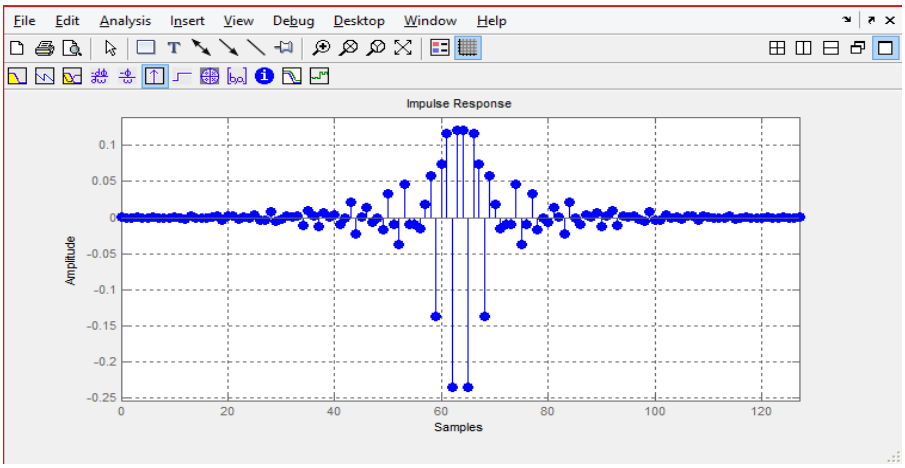
Fuente: elaboración propia, con programa MATLAB R2010b.

Figura 61. **Respuesta de fase filtro FIR**



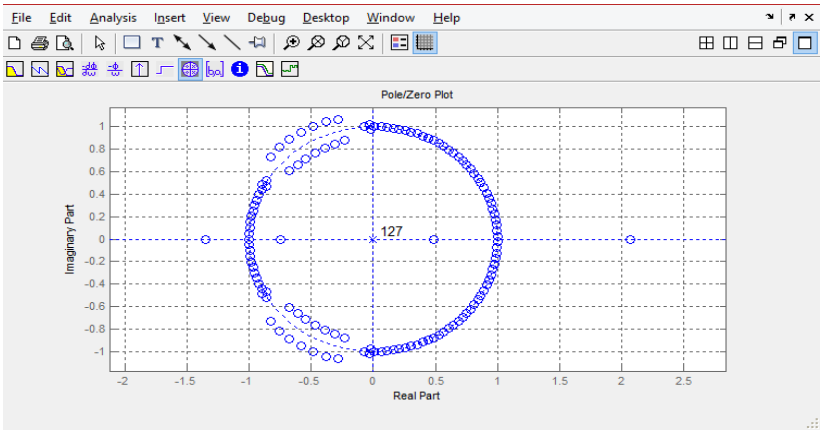
Fuente: elaboración propia, con programa MATLAB R2010b.

Figura 62. **Respuesta al impulso filtro FIR**



Fuente: elaboración propia, con programa MATLAB R2010b.

Figura 63. **Polos y ceros filtro FIR**

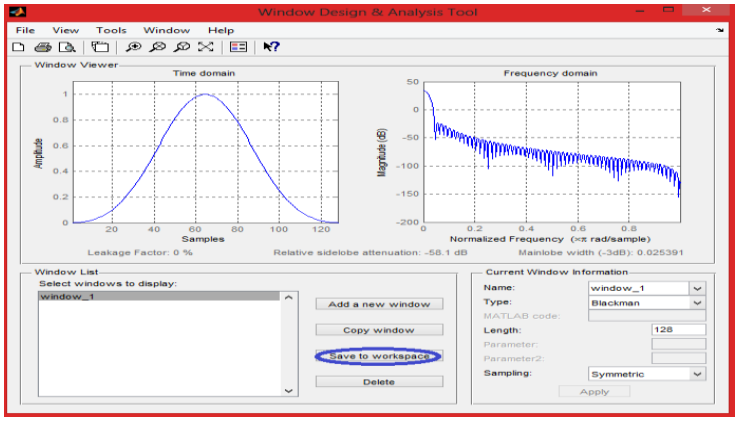


Fuente: elaboración propia, con programa MATLAB R2010b.

Se observa una serie de propiedades en las gráficas anteriores de todo el análisis que se puede hacer sobre un filtro utilizando esta herramienta *fvtool* al igual que se realizó en la práctica anterior con el filtro IIR.

Ahora se utiliza una herramienta que brinda MATLAB para el diseño de ventanas para filtros FIR que es *Wintool,* la cual es una GUI (interfaz gráfica). A continuación se observa cómo se utiliza. Ver figura 64.

Figura 64. **Interface gráfica Wintool**



Fuente: elaboración propia, con programa MATLAB R2010b.

Se diseña la función ventana *blackman* de longitud 128 y luego se exporta a MATLAB mediante la opción *Save to workspace.* El nombre por defecto es window_1.

Se realiza el filtro utilizando el resultado anterior de la siguiente manera. Ver figura 65.

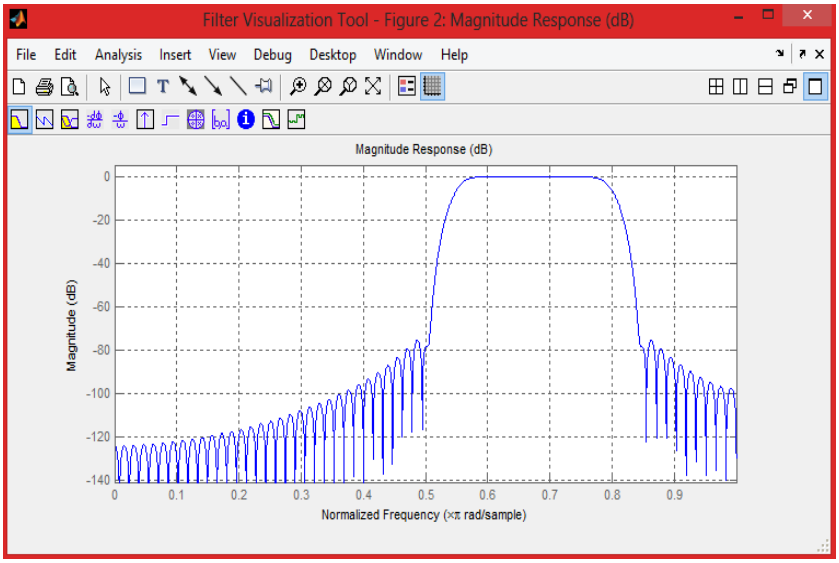
Figura 65. **Código fuente de la ventana *blackman***

```
window_1 has been exported to the workspace.
>> numBlackman=fir1(n,Wp,window_1);
>> fvtool(numBlackman)
```

Fuente: elaboración propia, con programa MATLAB R2010b.

Luego de ejecutar este comando, se observa el resultado nuevamente utilizando la herramienta fvtool ver figura 66.

Figura 66. **Respuesta en magnitud utilizando una ventana blackman**



Fuente: elaboración propia, con programa MATLAB R2010b.

Se observa que la diferencia con la respuesta en magnitud anterior sin utilizar WINTOOL, es que en la etapa de transición se obtiene una respuesta más acorde a la ideal.

Ahora como en la práctica anterior se utilizó el filtro diseñado para aplicarlo a un archivo de audio los comandos se observan en la figura 67.

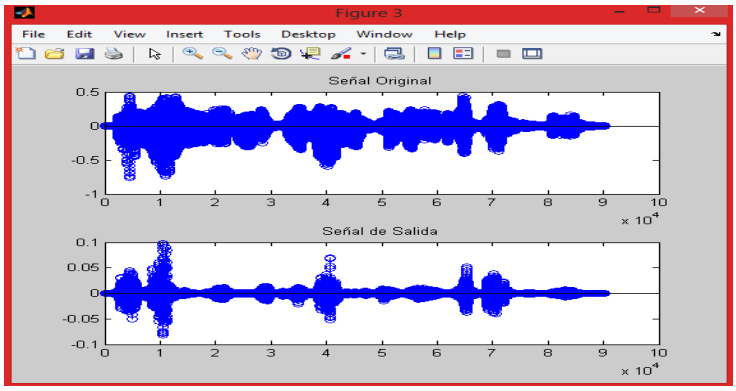
Figura 67. **Código fuente del filtro FIR aplicado a una señal de audio**

```
15
16 -    [xt,Fs]=wavread('x1');
17
18 -    yt=filter(numBlackman,1,xt);
19
20 -    soundsc(xt,Fs)
21 -    soundsc(yt,Fs)
22
23      %Gráficas:
24
25 -    subplot(2,1,1)
26 -    stem(xt)
27 -    title('Señal Original')
28
29 -    subplot(2,1,2)
30 -    stem(yt)
31 -    title('Señal de Salida')
32
```

Fuente: elaboración propia, con programa MATLAB R2010b.

Luego se ejecuta el código anterior y se observan las gráficas tanto de señal de entrada como de salida en la figura 68.

Figura 68. **Aplicando un filtro FIR a una señal de audio**



Fuente: elaboración propia, con MATLAB R2010b.

Se observa el cambio en la señal de audio al aplicarle un filtro, y a la hora de la práctica se podrá también escuchar el cambio entre la señal normal y la señal filtrada.

4.3.3. Recomendaciones

Realizar la práctica nuevamente utilizando otro tipo de opciones de ventanas que brinda MATLAB

Probar con diferentes tipos de audios los filtros, para tener una idea de hasta dónde puede llegar MATLAB y donde ya es necesario utilizar hardware.

5. DESARROLLO DE PRÁCTICAS DE LABORATORIO CON EL DSPIC30F4013

En este penúltimo capítulo se centra en la parte principal de este trabajo, diseñando 5 prácticas, las cuales se componen de dos partes principales, que son las siguientes:

- Construcción del hardware
- Implementación del software

Para el hardware se utilizó la placa entrenadora EasydsPIC4A, y para el software los programas copilados en lenguaje Basic que se encuentran en el apéndice de este trabajo.

5.1. Digitalización de señales analógicas con resolución de 12 bits

Esta práctica consta de digitalizar una señal analógica proveniente, ya sea del bloque ADC de la tarjeta entrenadora o de un generador de señales utilizando el A/D del dsPIC30F4013, trabajando con una profundidad de 12 bits. La información será enviada a un puerto B del dsPIC y se podrá observar la señal ya digitalizada en los LEDs de la tarjeta entrenadora EasydsPICA.

5.1.1. Objetivos

Ver lo práctico que es realizar la conversión de señales analógicas en señales digitales con la ayuda del dsPIC30F4013.

Manejar el convertidor interno ADC del dsPIC30F4013 con resolución de 12bits.

5.1.2. Recursos

Todo el hardware se encuentra en el Laboratorio de Comunicaciones 4, este equipo se tendrá que solicitar al auxiliar para cada práctica siguiendo las normativas del laboratorio, para esta práctica el equipo es el siguiente:

- Entrenadora EasydsPIC
- dsPIC30F4013
- Generador de Señales
- Datasheet dsPIC30F40013
- MicroBasic Pro for dsPIC30

5.1.3. Procedimiento

Paso 1: leer la teoría respecto al dsPIC30F4013 que se encuentra en este trabajo en la sección capítulo 3.

Paso 2: se arma el circuito de tal manera que la señal analógica sea conectada directamente a la entrada analógica del dsPIC30F4013 se utilizara el bloque ADC que se encuentra en la placa entrenadora o se puede utilizar un generador de señales con 1khz de frecuencia.

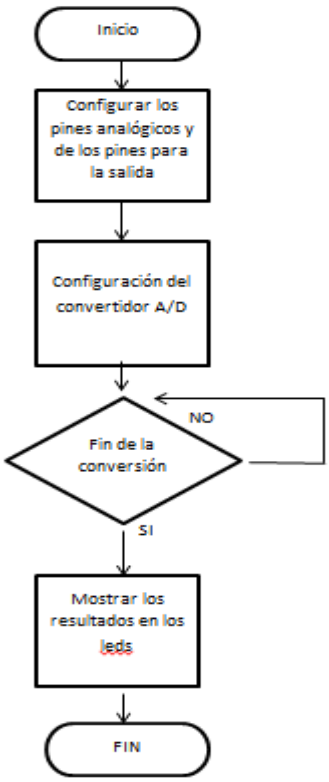
Paso 3: cargar al dsPIC el programa proporcionado en el apéndice de este trabajo.

Paso 4: por último describir los resultados que se observan en los LEDs de la placa entrenadora.

5.1.4. Diagrama de flujo del programa

A continuación en la figura 69 se muestran los pasos a seguir para la realización del programa para esta práctica de laboratorio, de esta manera es como está estructurado el programa que se encuentra en el apéndice A de este trabajo.

Figura 69. **Diagrama de flujo convertidor A/D**



Fuente: elaboración propia, con programa Microsoft Word 2010.

5.1.5. Resultados

Se puede observar en el bloque de LEDs de la placa entrenadora que para cada nivel de voltaje entre 0-5 v se asocia una palabra binaria con una resolución de 1,22 mv.

Se observa que cuando se tiene 5 v se encienden todos los LEDs y para 0v no se enciende ningún LED.

5.1.6. Recomendaciones

Realizar la misma práctica pero utilizando una señal de audio, y observar los resultados, teniendo en cuenta siempre el criterio de Nyquist.

Realizar la misma práctica solo que agregándole el module GLCD para poder observar en la pantalla el voltaje que se está aplicando al ADC.

5.2. Conversión D/A-red R-2R

En la siguiente práctica se realizó la conversión de una señal analógica, e inmediatamente la reconstrucción de la misma utilizando el método de R-2R, mediante este proceso se podrá observar el teorema de Nyquist, ya que si no se cumple el criterio de muestreo se tendrá una pérdida de la información, la cual la se podra ver reflejada en la salida como una señal de menor frecuencia.

Se digitalizan 3 señales analógicas variantes en el tiempo utilizando el A/D del dsPIC, trabajando como en la práctica anterior con una profundidad de 12 bits.

Las muestras tomadas se almacenaran en un vector Y del dsPIC y serán enviadas directamente al puerto del dsPIC, en donde se conectará a una red R-2R, la cual hará el trabajo de un D/C básico.

5.2.1. Objetivos

Realizar la conversión de una señal analógica en digital, utilizando las características del dsPIC30F4013 y la técnica de R-2R.

Conocer la técnica R-2R, para este tipo de aplicaciones y ver sus ventajas y desventajas de la misma.

5.2.2. Recursos

Todo el hardware se encuentra en el Laboratorio de Comunicaciones 4, este equipo se tendrá que solicitar al auxiliar para cada práctica siguiendo las normativas del laboratorio, para esta práctica el equipo es el siguiente:

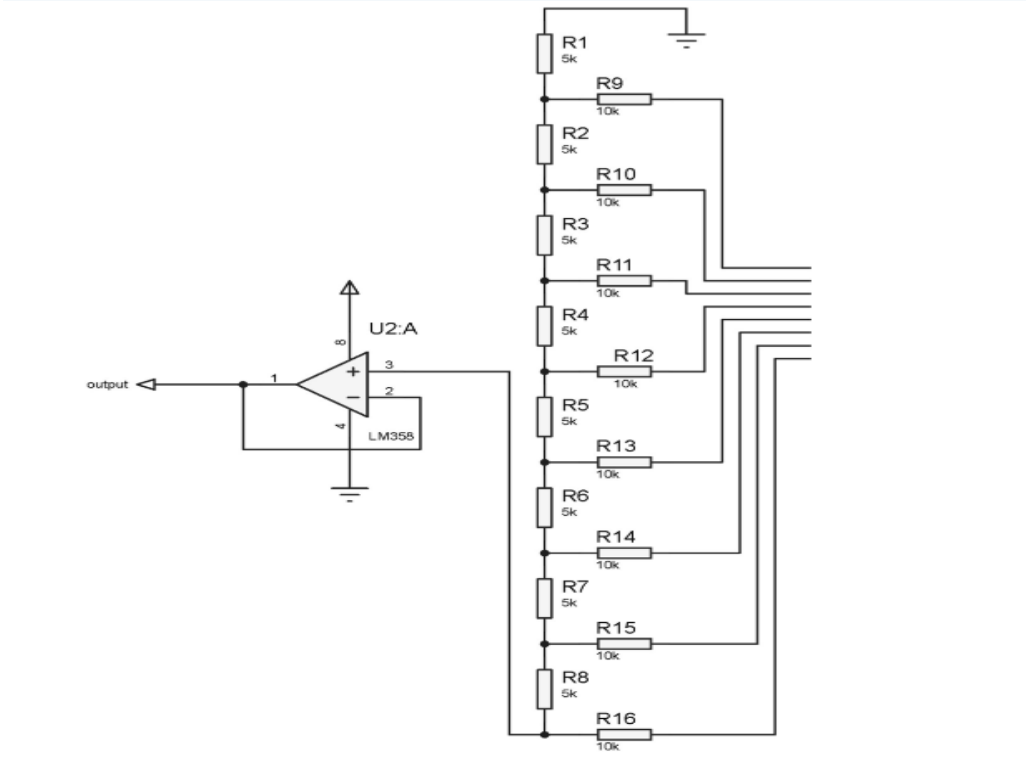
- Entrenadora EasydsPIC
- dsPIC30F4013
- Osciloscopio de 2 canales
- Red R-2R de 8 bits
- Generador de señales
- Datasheet dsPIC30F40013
- MicroBasic Pro for dsPIC30

5.2.3. Procedimiento

Paso 1: leer la teoría respecto a la técnica D/A-Red 2-2R en el capítulo 1.

Paso 2: se arma el circuito de tal manera que la señal analógica sea conectada directamente a la entrada analógica del dsPIC y la red 2-2R a la salida de un puerto, el canal 2 del osciloscopio debe ser conectado a esta última etapa ver figura 70 para el diagrama de la red 2-2R.

Figura 70. **Diagrama de la Red 2-2R**



Fuente: elaboración propia, con programa Multisim 13.0.

Paso 3: cargar al dsPIC el programa proporcionado en el apéndice de este trabajo (seria el mismo que para la práctica anterior).

Paso 4: realizar las pruebas con tres distintas señales, triangular, rectangular y una senoidal una a la vez, y verificar en el osciloscopio la correlación existente entre la señal de entrada y la señal de salida y anotar las observaciones.

5.2.4. Resultados

Se colocó una señal a la vez de 1 khz en la entrada del adc del dsPIC y ahí se coloca también el canal 1 del osciloscopio, luego se colocó el canal 2 en la última etapa después de la red 2-2R y se observó la salida, la cual tuvo que coincidir con la entrada.

Se observó que no sería exactamente igual, ya que en el proceso hubo ruido, distorsión por parte de cada etapa en el proceso además la red 2-2R no es la más recomendable para este tipo de aplicaciones, pero sin embargo con fines didácticos es una manera económica y sencilla de ver este proceso.

5.2.5. Recomendaciones

Para poder indagar más sobre el tema, y ver diferentes técnicas de conversión D/A se pueden ver otros métodos como utilizar el PWM del dsPIC.

Realizar la práctica utilizando audio en tiempo real, para poder observar el efecto este tipo de conversiones.

5.3. Diseño de un filtro IIR

Se digitalizó una señal analógica periódica en el tiempo atreves de A/D del dsPIC30f4013, trabajando con una profundidad de 12 bits a una tasa de muestreo constante. Las muestras se almacenaron en el vector Y del dsPIC y fueron procesadas por cada filtro diseñado en el dsPIC, almacenados en el vector X.

5.3.1. Objetivos

Diseñar un filtro IIR y conozca sus características principales.

Poner en práctica la teoría aprendida a lo largo de este trabajo.

5.3.2. Recursos

Todo el hardware se encuentra en el Laboratorio de Comunicaciones 4, este equipo se tendrá que solicitar al auxiliar para cada práctica siguiendo las normativas del laboratorio, para esta práctica el equipo es el siguiente:

- Entrenadora EasydsPIC
- Osciloscopio
- dsPIC30F4013
- mpc4921
- Generador de señales
- Datasheet dsPIC30F40013
- MicroBasic Pro for dsPIC30

5.3.3. Procedimiento

En esta práctica el código de programación fue generado por una herramienta de *MikroBasic* llamada *filter Designer Tool,* se describió paso a paso como diseñar el filtro utilizando esta herramienta se recomienda antes leer toda la teoría respecto a filtros digitales que se encuentra en este trabajo de graduación en el capítulo 1.

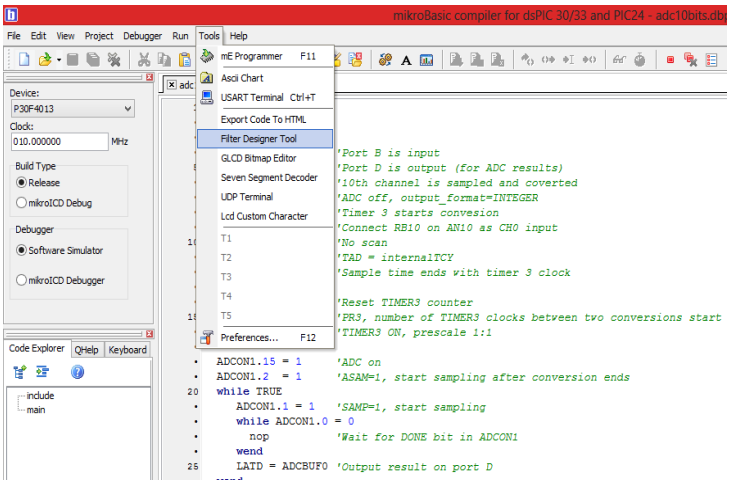
Se diseñará un filtro IIR con las siguientes características:

- Filtro pasa altas
- Ventana Butterworth
- Frecuencia de corte 2 kHz
- Frecuencia de muestreo 20 kHz.
- Filtro de orden n = 3

Los pasos a seguir para diseñar un filtro IIR son los siguientes:

Paso 1: abrir *filter Designer Tool,* que se encuentra en el menú *Tool* de *Mikrobasic*. Ver figura 71.

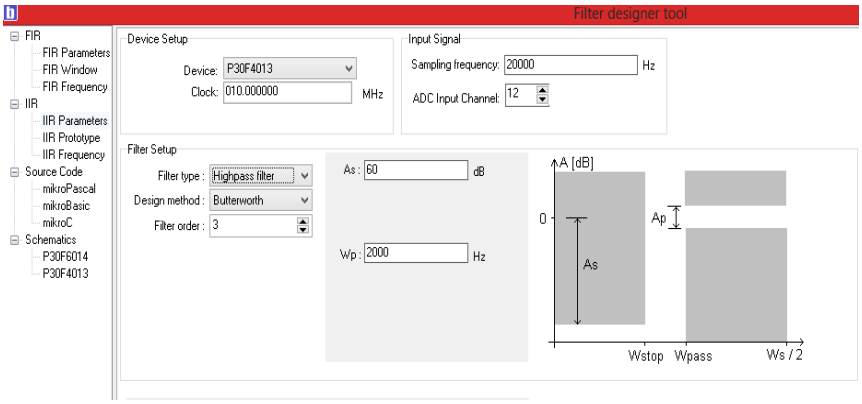
Figura 71. **Interface *filter Designer Tool***



Fuente: *mikroBasic compiler for dsPIC* v 5.0.0.0.

Paso 2: configurar el filtro según el diseño propuesto anteriormente. Ver figura 72.

Figura 72. **Interface gráfica de filter Designer Tool**

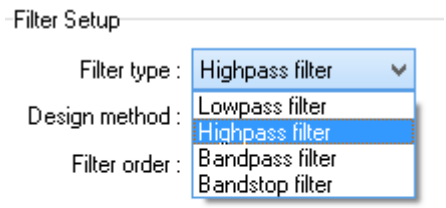


Fuente: *mikroBasic compiler for dsPIC* v 5.0.0.0.

En esta ventana se seleccionan las diferentes opciones para la configuración del filtro, se observa en el menú de lado derecho que tiene la opción de seleccionar si quiere un filtro FIR o un filtro IIR, para esta práctica se seleciona la opción IIR.

En esta ventana es donde se configura el filtro, se seleciona el dsPIC a utilizar, la frecuencia de muestreo, tipo de filtro, el método de diseño, el orden del filtro y ahora se tiene una característica importante que es la amplificación del filtro la cual se mide en dB se deja el que trae por defecto, ver figuras 73,74 y 75.

Figura 73. **Selección del tipo de filtro**

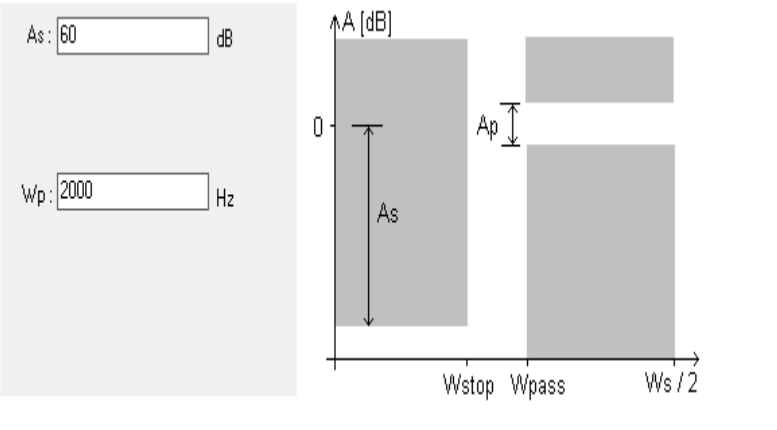


Fuente: mikroBasic compiler for dsPIC v 5.0.0.0.

Figura 74. **Selección de frecuencia de muestreo**

Fuente: mikroBasic compiler for dsPIC v 5.0.0.0.

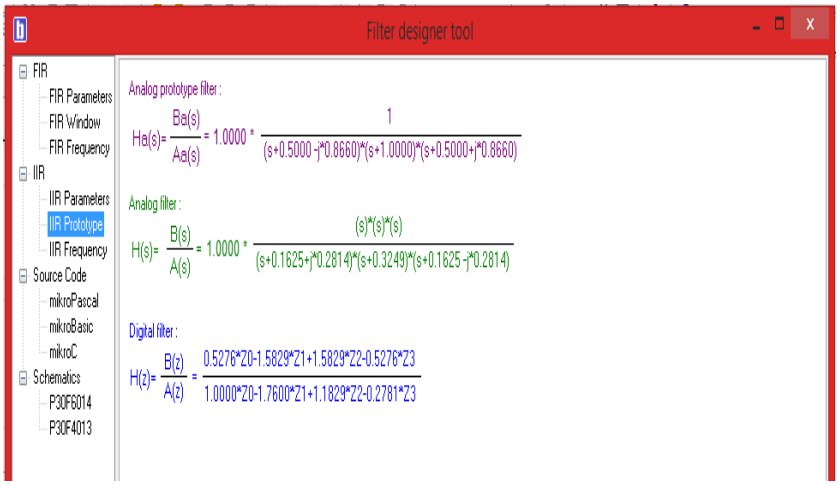
Figura 75. **Selección de frecuencia de corte**



Fuente: mikroBasic compiler for dsPIC v 5.0.0.0.

Paso 3: ahora se observa una ventada donde se muestra la función de transferencia del filtro diseñado anteriormente. Ver figura 76.

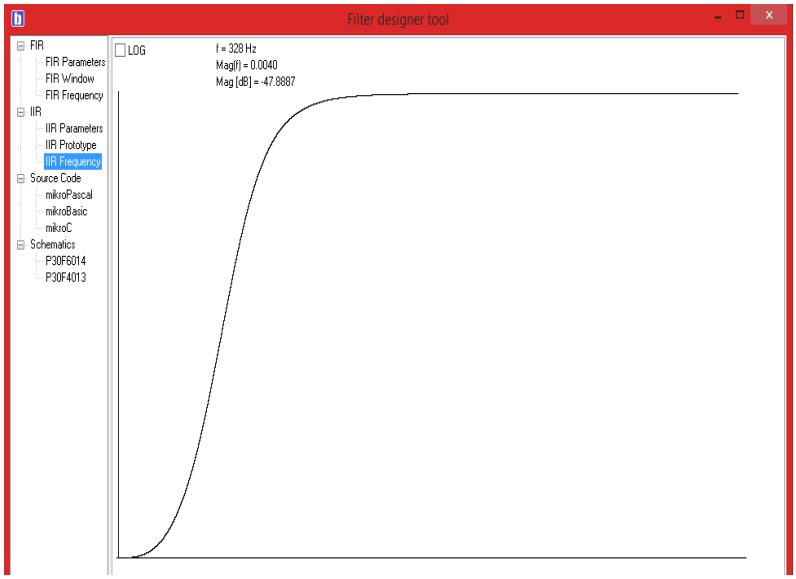
Figura 76. **Función de transferencia de un filtro IIR**



Fuente: elaboración propia, con programa mikroBasic compiler for dsPIC v 5.0.0.0.

Paso 4: ahora se observa la gráfica de respuesta en frecuencia que brinda *filter Designer Tool,* ver figura 77.

Figura 77. **Respuesta en frecuencia de un filtro IIR**

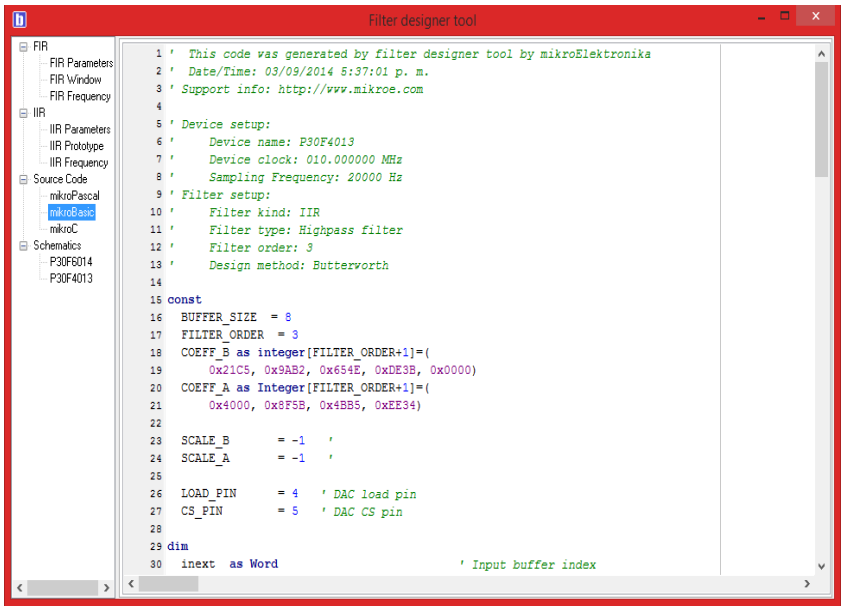


Fuente: elaboración propia.

Se observa en la figura 77, que efectivamente es un filtro pasa altas y que cumple con las especificaciones dadas.

Pasó 5: por último se genera el código para el compilador que es Mikrobasic de la siguiente manera. Ver figura 78.

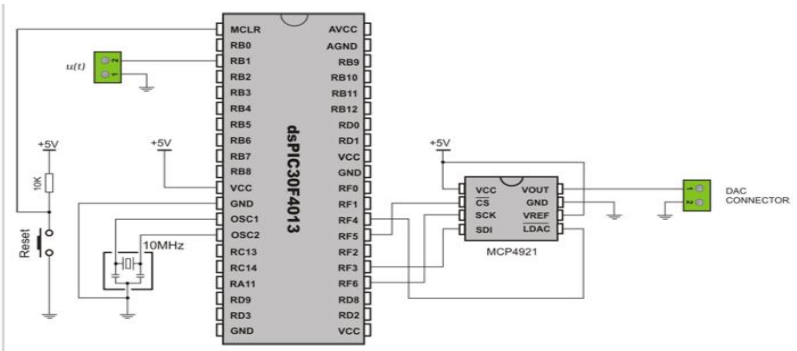
Figura 78. **Código fuente filtro IIR**



Fuente: elaboración propia.

Paso 6: Se descarga el programa en el dsPIC y se arma el circuito ver figura 79.

Figura 79. **Diagrama de conexión del dsPIC30F4013**



Fuente: *mikroBasic compiler for dsPIC* v 5.0.0.0.

En el esquema anterior muestra como conectar el DAC MCP4921 al dsPIC30F4013, básicamente el DAC se conectará al SPI del dsPIC.

Paso 7: ahora ya se realizan pruebas con el filtro, utilizando el generador de señales, un voltaje de 1v y onda senoidal, se hace variar de frecuencia y observar mediante el osciloscopio la salida.

5.3.4. Resultados

Se observó que a frecuencias menores de 2 Khz el resultado en la salida del dsPIC fue nulo, mientras que para frecuencias mayores a 2 khz se obtendrá una señal en la salida, estos valores son fáciles de modificar, solo es de seguir los pasos anteriormente propuestos.

5.3.5. Recomendaciones

Probar con varios valores de frecuencia de corte, para así poder observar mejor el funcionamiento del filtro.

Probar con diferentes valores de amplificación, que permite el programa, y así poder analizar cuál sería el más óptimo para esta práctica.

5.4. Diseño de un filtro FIR

Esta práctica es similar a la anterior con la única variedad que ahora el diseño estuvo basado en un filtro FIR, así podrá observar la deferencia entre uno y otro.

5.4.1. Objetivos

Que el estudiante aprenda a diseñar filtros FIR, utilizando la herramienta *filter Designer Tool* proporcionada por MikroBasic.

Que el estudiante diseñe filtros FIR pasa bajos por diferentes métodos y ver diferencias con los filtros IIR.

5.4.2. Recursos

Todo el hardware se encuentra en el Laboratorio de Comunicaciones 4, , este equipo se tendrá que solicitar al auxiliar para cada práctica siguiendo las normativas del laboratorio, para esta práctica el equipo es el siguiente:

- Entrenadora EasydsPIC
- Osciloscopio
- dsPIC30F4013
- mpc4921
- Generador de señales
- Datasheet dsPIC30F40013
- MicroBasic Pro for dsPIC30

5.4.3. Procedimiento

En esta práctica al igual que la anterior, el código de programación será generado por una herramienta de MikroBasic llamada *filter Designer Tool,* mostrare paso a paso como diseñar el filtro utilizando esta herramienta se recomienda antes leer toda la teoría respecto a filtros digitales que se encuentra en este trabajo en el capítulo 1.

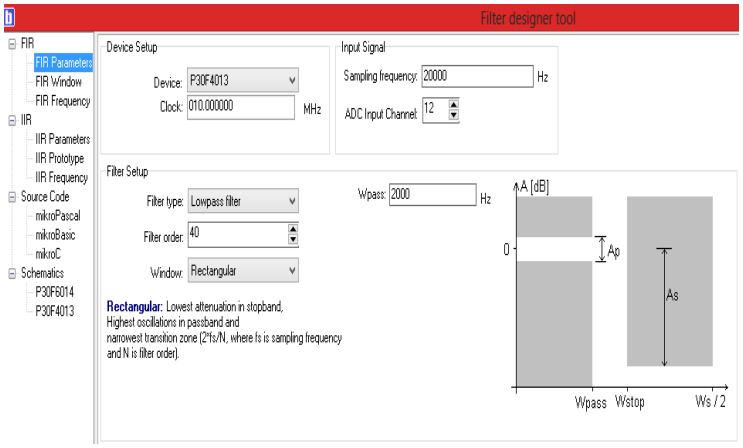
Se diseñará un filtro FIR con las siguientes características:

- Filtro pasa bajos
- Ventana rectangular
- frecuencia de corte 2kHz
- frecuencia de muestreo 20kHz.
- Filtro de orden n=40

Paso 1: abrir *filter Designer Tool,* como lo hicimos en la práctica anterior.

Paso 2: después de darle clic, abrirá la siguiente ventada ver figura 80.

Figura 80. **Interface gráfica de Filter Designer Tool**



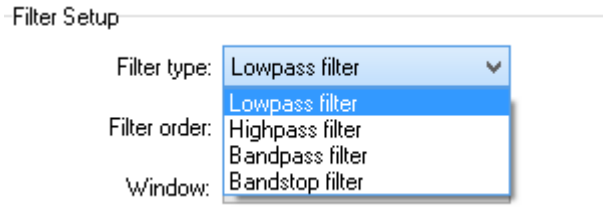
Fuente: *mikroBasic compiler for dsPIC* v 5.0.0.0.

En esta ventada se seleccionan las diferentes opciones para la configuración del filtro, se observa en el menú de lado derecho que se tiene la opción de seleccionar si se quiere un filtro FIR o un filtro IIR, para esta práctica se selecciona la opción FIR.

Luego se selecciona el dispositivo a utilizar, así como la frecuencia del cristal, en este caso se selecciona entre los dispositivos que se muestra, el dsPIC30F4013 y la frecuencia del cristal en 10 khz que es el que se ha estado utilizando.

Pasó 3: luego se seleciona el tipo de filtro, en este caso es paso bajo, ver figura 81.

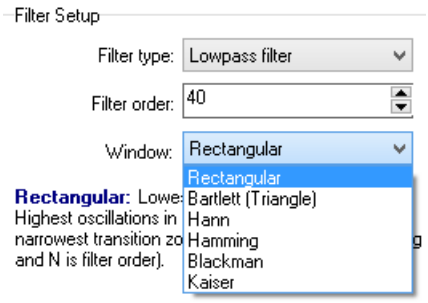
Figura 81. **Selección de tipo de filtro**



Fuente: *mikroBasic compiler for dsPIC* v 5.0.0.0.

Paso 4: ahora se selecciona el orden y el tipo de ventana con el que se va a trabajar, ver figuras 82.

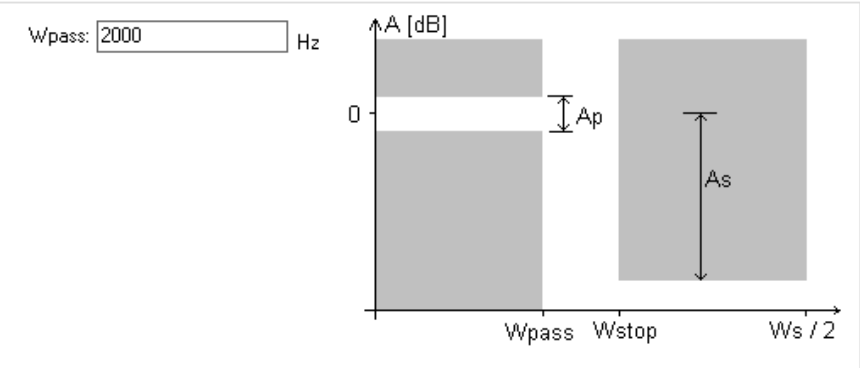
Figura 82. **Selección de tipo de ventana**



Fuente: *mikroBasic compiler for dsPIC* v 5.0.0.0.

Paso 5: por último se seleciona la frecuencia de corte en esta práctica es de 2 000 Hz, y se observa un diagrama de las características del filtro. Ver figura 83.

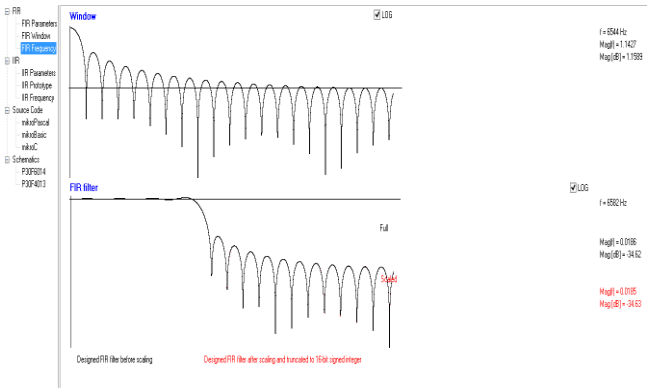
Figura 83. **Selección de frecuencia de corte**



Fuente: *mikroBasic compiler for dsPIC* v 5.0.0.0.

Pasó 6: en el menú de lado izquierdo de la pantalla principal de *filter Designer Tool,* se selecciona la opción Fir Frequency, se observará un diagrama con las características de repuesta en frecuencia del filtro. Ver figura 84.

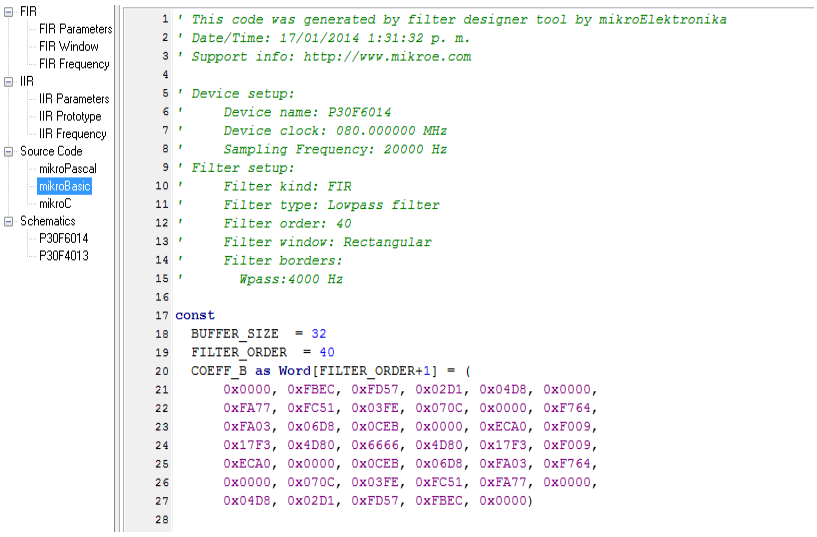
Figura 84. **Respuesta en frecuencia filtro FIR**



Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Paso 7: se genera el código, como se realizó en la práctica anterior.

Figura 85. **Código fuente filtro FIR**

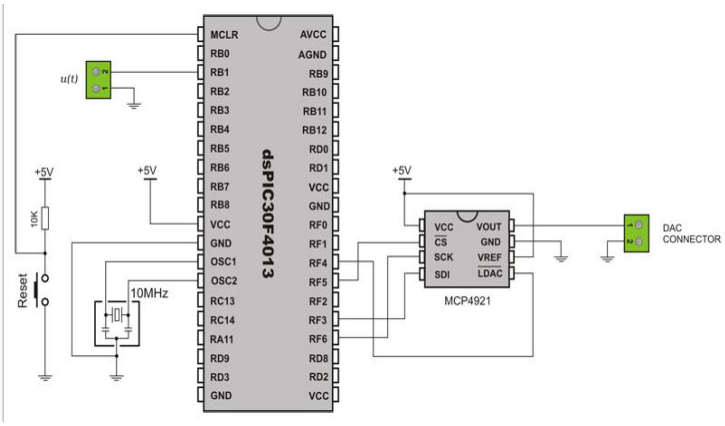



```
' This code was generated by filter designer tool by mikroElektronika
' Date/Time: 17/01/2014 1:31:32 p. m.
' Support info: http://www.mikroe.com

' Device setup:
'     Device name: P30F6014
'     Device clock: 080.000000 MHz
'     Sampling Frequency: 20000 Hz
' Filter setup:
'     Filter kind: FIR
'     Filter type: Lowpass filter
'     Filter order: 40
'     Filter window: Rectangular
'     Filter borders:
'       Wpass:4000 Hz

const
  BUFFER_SIZE  = 32
  FILTER_ORDER  = 40
  COEFF_B as Word[FILTER_ORDER+1] = (
      0x0000, 0xFBEC, 0xFD57, 0x02D1, 0x04D8, 0x0000,
      0xFA77, 0xFC51, 0x03FE, 0x070C, 0x0000, 0xF764,
      0xFA03, 0x06D8, 0x0CEB, 0x0000, 0xECA0, 0xF009,
      0x17F3, 0x4D80, 0x6666, 0x4D80, 0x17F3, 0xF009,
      0xECA0, 0x0000, 0x0CEB, 0x06D8, 0xFA03, 0xF764,
      0x0000, 0x070C, 0x03FE, 0xFC51, 0xFA77, 0x0000,
      0x04D8, 0x02D1, 0xFD57, 0xFBEC, 0x0000)

```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Paso 8: ahora ya se tiene generado el código, se descarga al dsPIC y se arma el circuito ver figura 86.

Figura 86. **Diagrama de conexión para el filtro FIR**



Fuente: *mikroBasic compiler for dsPIC* v 5.0.0.0.

Paso 9: ahora ya se realizan pruebas con el filtro, utilizando el generador de señales, un voltaje de 1v y onda senoidal, se hace variar la frecuencia y se observa mediante el osciloscopio la salida.

5.4.4. Resultados

Se observó que a frecuencias mayores de 2 kilohertz, se tendrá una salida nula, esto se debe al diseño de este filtro, estas configuraciones son fáciles de modificar siguiendo la guía antes propuesta.

5.4.5. Recomendaciones

Realizar la misma práctica solo que modificando los valores como lo son frecuencia de muestreo, de corte, orden del filtro y tipo de filtro.

En cualquier diseño que se haga siempre tomar en cuenta que la frecuencia de muestreo cumpla con el criterio de Nyquist.

5.5. Transformada de Fourier

La práctica consta en poder calcular o interpretar la transformada de Fourier *fft* de una señal analógica con conceptos básicos. La señal de entrada es muestreada en intervalos regulares y almacenados en un *búfer*.

Luego que el *búfer* está lleno, esto es pasado a un subproceso de la *fft*, aquí es calculada la rápida (discreta) transformación de Fourier de la señal de entrada y vuelve de nuevo las muestras de la *fft* en el buffer de entrada.

La amplitud de cada muestra de la *fft* es calculada como *f[k]=sqrt(re[k]^2+im[k]^2).* La amplitud del espectro es dibujado en la GLCD. Todo el proceso se repite infinitamente.

5.5.1. Objetivos

Realizar la transformada de Fourier de una señal analógica, cuadrada y triangular.

Utilizar el módulo Glcd de la placa entrenadora y darle una aplicación práctica.

5.5.2. Recursos

Todo el hardware se encuentra en el Laboratorio de Comunicaciones 4, este equipo se tendrá que solicitar al auxiliar para cada práctica siguiendo las normativas del laboratorio, para esta práctica el equipo es el siguiente:

- Entrenadora EasydsPIC
- dsPIC30F4013
- Generador de señales
- Datasheet dsPIC30F40013
- MicroBasic Pro for dsPIC30
- GLCD 128X64

5.5.3. Procedimiento

Paso 1: cargar al dsPIC el programa proporcionado en el apéndice de este trabajo llamado, *transformada de Fourier.*

Paso 2: como en la tarjeta entrenadora ya se encuentra conectada la GLCD al dsPIC, solo queda revisar que la tarjeta esté conectada a la fuente de poder.

Paso 3: conectar el generador se a la entrada analógica RB8 del dsPIC.

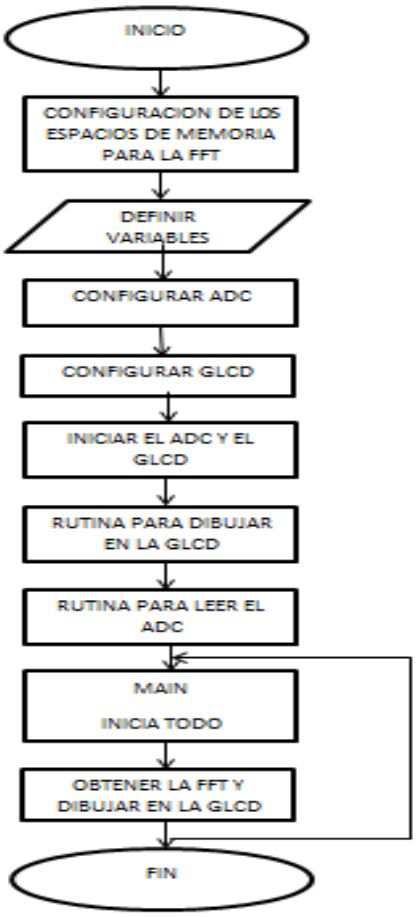
Paso 4: realizar las pruebas siguientes:

- Onda senoidal con 1 400 Hz
- Onda cuadrada con 1 200 Hz
- Onda triangular con 1 400 Hz

5.5.4. Diagrama de flujo del programa

A continuación se muestran los pasos a seguir para la realización del programa para esta práctica de laboratorio. Ver figura 87. De esta manera es como está estructurado el programa que se encuentra en el apéndice C de este trabajo.

Figura 87. **Diagrama de flujo transformada de Fourier**



Fuente: elaboración propia, con programa Microsoft Word 2010.

5.5.5. Resultados

Con las pruebas realizadas anteriormente descrita, se observa en la GLCD el espectro de amplitud correspondiente a cada onda aplicada y la frecuencia de dicha onda.

Si se experimentan extraños dibujos en la GLCD se debe intentar alterar los niveles de la señal de entrada hasta que se logre estabilizar la imagen.

5.5.6. Recomendaciones

Revisar el generador de señales en cada prueba que se realice para que la frecuencia no exceda los 6 kilohertz y en todo momento la amplitud debe ser de 1 voltio.

Realizar la práctica con diferentes señales a las descritas en esta práctica y observar los resultados para observar el alcance del algoritmo anteriormente planteado.

6. OTRAS APLICACIONES CON EL DSPIC

Este último capítulo trata de 3 prácticas que no tienen relación directamente con el tema principal de este trabajo, pero es igualmente importante conocer estas aplicaciones y como trabajarlas con el dsPIC y así el estudiante podrá tener una visión más amplia sobre este microcontrolador.

6.1. Controlando un motor C.C. mediante PWM

En esta práctica se describe como emplear la capacidad del dsPIC para generar señales PWM y regular la velocidad de funcionamiento de un motor c.c., se configurará el módulo OC1 del dsPIC30F4013 para generar una señal cuadrada que se aplica a un chip amplificador que regule la velocidad del motor.

6.1.1. Objetivos

Controlar un motor de corriente continua con un dsPIC de propósito general como los es el dsPIC30F4013.

Utilizar el módulo OC1 del dsPIC30F4013 y poder generar una señal PWM.

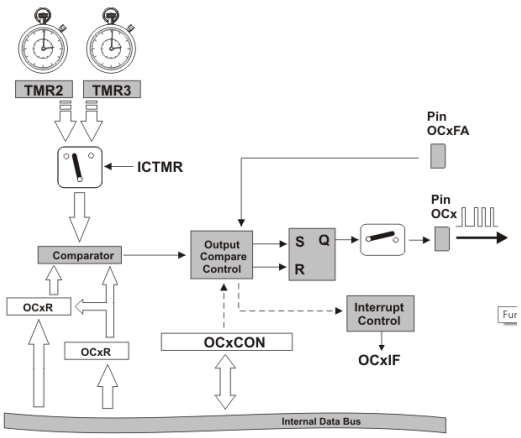
6.1.2. Descripción de la práctica

A continuación se procede a regular la velocidad de un motor de corriente continua con 4 velocidades distintas, que se podrá seleccionar mediante 2 interruptores de la tarjeta entrenadora Easypic.

Con la combinación 00, el motor está apagado e irá aumentando su velocidad progresivamente hasta que se seleccione el máximo valor binario correspondiente a los 2 interruptores activados en 11.

La práctica emplea el módulo hardware *Output compare* del dsPIC configurado en modo PWM sencillo, para realizar el control de este tipo de motor se fundamenta en la tensión de alimentación del motor. Cuando mayor sea la tensión en los bornes del motor, mayor será el giro de su eje. A continuación, se observa un diagrama de cómo es que funciona esta característica en el dsPIC30F4013. Ver figura 88.

Figura 88. **Diagrama Output compare Module**



Fuente: http://www.mikroe.com/chapters/view/39/chapter-6-output-compare-module. Consulta: 25 de mayo de 2014.

6.1.3. Recursos

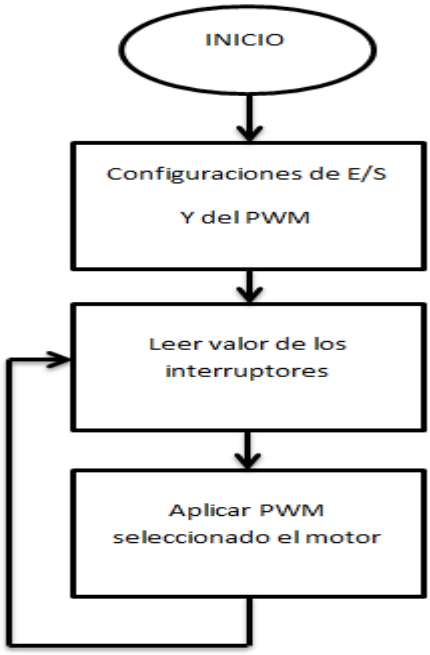
Todo el hardware se encuentra en el Laboratorio de Comunicaciones 4, este equipo se tendrá que solicitar al auxiliar para cada práctica siguiendo las normativas del laboratorio, para esta práctica el equipo es el siguiente:

- Entrenadora EasydsPIC
- dsPIC30F4013
- El driver de motores L293D
- Motor DC
- Datasheet dsPIC30F40013

6.1.4. Diagrama de flujo del programa

A continuación se muestran los pasos a seguir para la realización del programa para esta práctica de laboratorio, ver figura 89, de esta manera es como está estructurado el programa que se encuentra en el apéndice I de este trabajo.

Figura 89. **Diagrama de flujo PWM**



Fuente: elaboración propia, con programa Microsoft Word 2010.

6.1.5. Resultados

Como en esta práctica se usa un *driver* de motores L293D como interfaz entre el dsPIC y el motor, dicho *driver* no se encuentra en la tarjeta entrenadora tendrá que trabajarse en un protoboar para poder realizar las conexiones adecuadamente. El diagrama de cómo realizar las coneciones se puede encontrar en la web.

Al momento de realizar las combinaciones lógicas, 00, 01,10 y 11, con los *switches* conectados a los pines RF0 y RF1 se podrá observar el aumento de velocidad del motor cc. De igual manera realizar las pruebas de forma ascendente y descendente para poder observar mejor el fenómeno.

6.1.6. Recomendaciones

Indagar más sobre esta característica con la que cuenta este tipo de dsPIC que si bien no es su especialidad, puede realizar bien el trabajo a una escala menor.

Realizar otro tipo de prácticas utilizando el hardware Output compare del dsPIC para así conocer más a fondo sobre esta característica.

6.2. Frecuencímetro

En esta práctica se realizará una herramienta muy útil pero a la vez sencilla, ya que es con fines didácticos, se utiliza lo visto anteriormente y se verá con más detalle cómo utilizar los *timer* del dsPIC, que es la base principal para la realización de esta práctica.

6.2.1. Objetivos

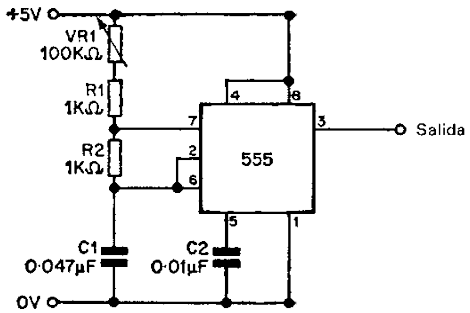
Diseñar un circuito, el cual permita por medio de un bloque de 7 segmentos visualizar la frecuencia de 1-50 Hz.

Que el estudiante pueda manejar los *timers* del dsPIC para futuras implementaciones.

6.2.2. Procedimiento

Paso 1: armar el circuito 555 de la figura en un *protoboar*. Ver figura 90.

Figura 90. **Diagrama de un circuito 555**



Fuente: http://www.electronicafacil.net/circuitos/Multivibrador-astable-con-555.html. Consulta: 11 de junio de 2014.

Paso 2: se conecta la salida del 555 al puerto D del del dsPIC.

Paso 3: se descarga el programa de esta práctica, que se encuentra en el apéndice de este trabajo.

Paso 4: se conecta el bloque de 7 segmentos al puerto B del dsPIC y se observan los resultados.

6.2.3. Recursos

Todo el hardware se encuentre en el laboratorio de comunicaciones 4, este equipo se tendrá que solicitar al auxiliar para cada práctica siguiendo las normativas del laboratorio, para esta práctica el equipo es el siguiente:

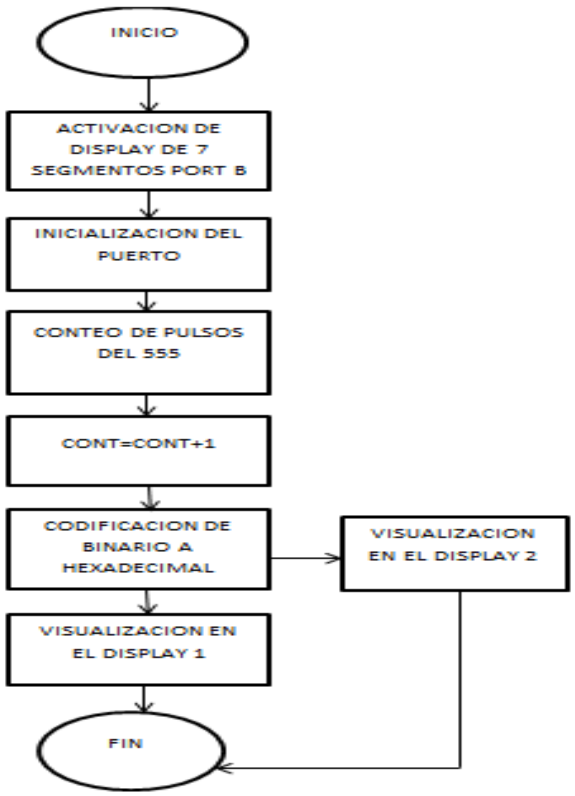
- Entrenadora EasydsPIC
- dsPIC30F4013
- circuito 555

- Datasheet dsPIC30F4013
- Bloque de 7 segmentos

6.2.4. Diagrama de flujo

A continuación se muestran los pasos a seguir para la realización del programa para esta práctica de laboratorio. Ver figura 91. De esta manera es como está estructurado el programa que se encuentra en el apéndice J de este trabajo.

Figura 91. **Diagrama de flujo frecuencímetro**



Fuente: elaboración propia, con programa Microsoft Word 2010.

6.2.5. Resultados

Se observa cómo irá variando la lectura en los *displays* conforme se va variando la frecuencia en el 555.

6.2.6. Recomendaciones

Esta práctica se hizo con una frecuencia de 0-50 Hertz, como mejor se podría utilizar otro circuito o algún integrado que genera un rango más amplio de frecuencia, ya que se conoce de las grandes capacidades con las que cuenta un dsPIC.

Realizar nuevamente la práctica, solo que ahora utilizando; ya sea la lcd o glcd para poder visualizar la frecuencia.

6.3. Comunicación encriptada

En esta práctica se propone comunicar el dsPIC30F4013 con una computadora por medio del puerto serie de una PC, se utilizará el hardware USART disponible en la placa entrenadora. Para poder observar los datos enviados se puede utilizar cualquier programa de *Hyperterminal* de comunicaciones, en este caso se utiliza el que trae por defecto Windows o el de *mikrobasic*.

Se utilizará las características matemáticas del dsPIC para poder realizar una encriptación, ya que se pueden realizar operaciones más complejas y de una forma rápida que con un simple MCU, no se podría hacer.

En esta práctica se envia un dato que previamente será elevado al cuadrado y dividido entre 10, tras realizar esta operación da un cociente y un residuo, ambos de 8 bits.

6.3.1. Objetivos

Utilizar la capacidad del dspic para la encriptación de datos en las comunicaciones.

Configurar y conocer los comandos del dsPIC para poder realizar la comunicación con la computadora mediante el puerto serie.

6.3.2. Recursos

Todo el hardware se encuentra en el Laboratorio de Comunicaciones 4, , este equipo se tendrá que solicitar al auxiliar para cada práctica siguiendo las normativas del laboratorio, para esta práctica el equipo es el siguiente:

- Entrenadora EasydsPIC
- dsPIC30F4013
- Mickrobasic
- Datasheet dsPIC30F40013

6.3.3. Procedimiento

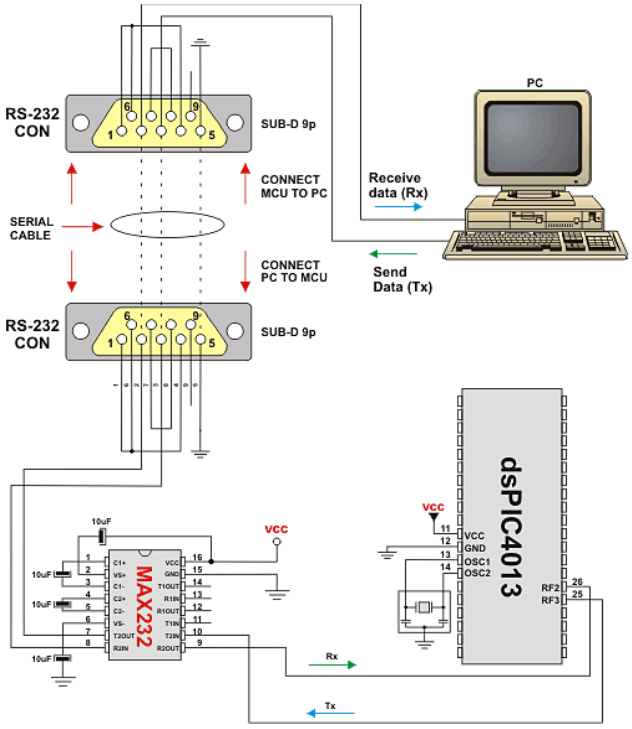
Paso 1: leer la teoría correspondiente a este tema que se encuentra en el capítulo 3.

Paso 2: copilar y grabar el programa al dsPIC utilizando la entrenadora, dicho programa se encuentra en el apéndice de este trabajo.

Paso 3: revisar que las conexiones estén bien en la placa entrenadora. Ver figura 92.

Paso 4: realizar las pruebas para ver si todo funciona bien.

Figura 92. **Diagrama de conexión del USAR**

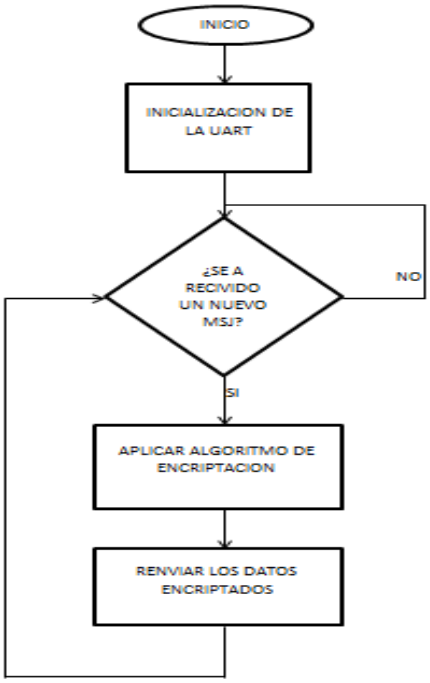


Fuente: http://www.mikroe.com/chapters/view/43/chapter-10-uart-module/.
Consulta: 2 de julio de 2014.

6.3.4. Diagrama de flujo

A continuación se muestran los pasos a seguir para la realización del programa para esta práctica de laboratorio. Ver figura 93. De esta manera es como está estructurado el programa que se encuentra en el apéndice N de este trabajo.

Figura 93. **Diagrama de flujo comunicación encriptada**

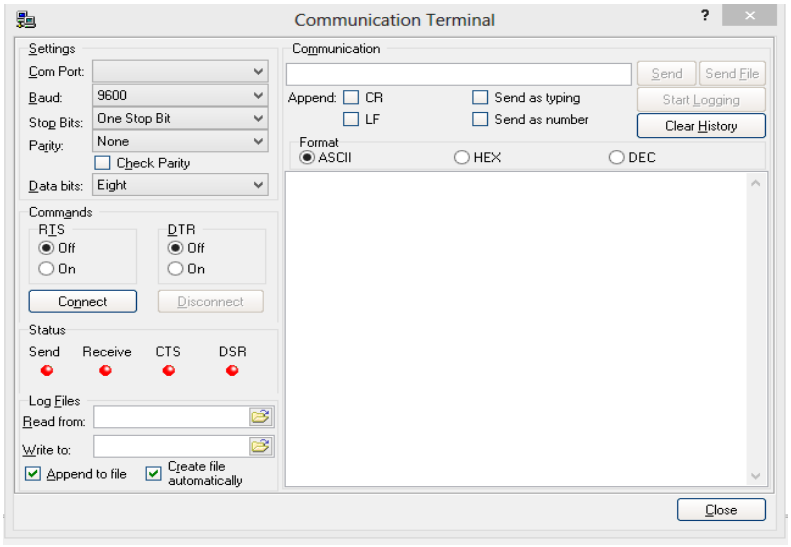


Fuente: elaboración propia, con programa Microsoft Word 2010.

6.3.5. Resultados

Se procede a abrir el hyperterminal de mikrobasic, esto se encuentra en el menú tool->Usart terminal, y se observa una ventana como en la figura 94.

Figura 94. **Ventada de interface hypeterminal**



Fuente: *mikroBasic compiler for dsPIC* v 5.0.0.0.

Todo está ya configurado por defecto como se puede observar, y por último se mandan datos a través del *hyperterminal* y poder observar en la lcd de la tarjeta entrenadora.

6.3.6. Recomendaciones

Se pueden utilizar varias formas de encriptación para poder aprovechar más los recursos matemáticos del dsPIC30F4013.

Utilizar también la Glcd para poder tener una mejor visualización de la palabra codificada en incluso mejorar la práctica utilizando este periférico.

CONCLUSIONES

1. Utilizando la transformada de Fourier y la transformada Z, se logra interactuar con las diferentes señales para entender su comportamiento, tanto en tiempo continuo como en el tiempo discreto.

2. MATLAB es un software de gran capacidad para el procesamiento de señales digitales, con el cual se logró diseñar filtros tanto IIR como FIR.

3. El dsPIC30F4013 es un microcontrolador de alta capacidad para el desarrollo de aplicaciones en el tema de procesamiento de señales digitales.

4. Se logró diseñar 3 prácticas utilizando el software MATLAB para que el estudiante conozca este software y le sea útil en otros cursos de la carrera.

5. Se realizó el diseño de 5 prácticas utilizando el dsPIC30F4013, con esto el estudiante logrará desarrollar habilidad en la utilización y programación de dicho microcontrolador.

6. Se logró proponer 3 prácticas más, en donde se desarrolla otras aplicaciones de los dsPICs, que serán de gran ayuda para futuros proyectos.

RECOMENDACIONES

1. Motivar a los estudiantes a seguir investigando nuevas prácticas o desarrollo de proyectos utilizando MATLAB, ya que es un software con muchas capacidades para diversas aplicaciones en el campo de la electrónica.

2. Elaborar proyectos utilizando el dsPIC, ya que es un microcontrolador de bajo costo y de muy amplias capacidades en diversas aplicaciones en el campo de la electrónica, especialmente en el procesamiento de señales digitales.

3. Utilizar este trabajo como una guía para el desarrollo prácticas de Laboratorio de Comunicaciones 4, tanto para MATLAB como para el dsPIC304013.

4. Ir actualizando las prácticas que aquí se encuentran o ir agregando nuevas, ya que la electrónica es un mundo en continuo cambio y los estudiantes tienen que ir al ritmo de este.

5. Establecer un curso más formal como lo hay para PICs, para los dsPICs, ya que hay herramienta en Laboratorio de Comunicaciones 4, como placas entrenadoras y microcontrolador para que es estudiante desarrollo más su habilidad de programación en este tipo de dispositivos.

BIBLIOGRAFÍA

1. ANGULO USATEGUI, Jose. *dsPIC diseño práctico de aplicaciones*. España: McGrawHall, 2008. 351 p.

2. *Conversor AD del dsPIC30F4013.* [en línea]: <http://server-die.alc.upv.es/asignaturas/PAEEES/2006-07/Conversor%20AD%20del%20dsPIC30F4013.pdf> [Consulta: 20 de agosto de 2013.]

3. *Conversores D/A y A/D*. [en línea]: <http://www.fceia.unr.edu.ar/enica3/da-ad> [Consulta: 15 de junio de 2013.]

4. *Convertidor DAC.* [en línea]: <http://proton.ucting.udg.mx/~sanchez/EL%20DAC.htm> [Consulta: 2 de junio de 2013.]

5. *Data sheet dsPIC30F4013*. [en línea]: <http://ww1.microchip.com/downloads/en/devicedoc/70138c.pdf.> [Consulta: 25 de febrero de 2014.]

6. Hsu hwei p. *Analisis de Fourier*. México: Prentice Hall, 1998. 269 p.

7. *La transformada z*. [en línea]: <http://dctrl.fib.unam.mx/ricardo/Transformada%20Z/La%20Transformada%20Z_corregido> [Consulta: 13 de mayo de 2013.]

8. MikroElectronica. *User Manual of EasydsPIC4A*. [en línea]: <http://www.mikroe.com/downloads/get/333/easydspic4a_manual_v100> [Consulta: 16 de julio de 2013.]

9. PLATERO, Carlos. *Introducción al procesamiento digital de señales*. España: Universidad Politécnica de Madrid, 2003. 760 p.

10. *Procesamiento de señales digitales*. [en línea]: <http://www.angelfire.com/falcon/shadow_rsv/t_senales/curdsp1.pdf> [Consulta: 8 de abril de 2013.]

11. *Procesamiento digital de señales con MATLAB*. [en línea]: <http://es.scribd.com/doc/46935329/PROCESAMIENTO-DIGITAL-DE-SENALES-CON-MATLAB> [Consulta: 12 de marzo de 2014]

12. *Señales*. [en línea]: <http:www.euv.cl/archivos_pdf/señales.pdf> [Consulta: 2 de mayo de 2013.]

APÉNDICES

En este apartado se encuentran los programas necesarios para realizar las prácticas que son los siguientes.

Apéndice A. **Código fuente práctica conversión 12 bits parte I**

```
dim num as word
dim lectura as word
dim cont as word [256]

sub procedure leer
    lectura =ADC1_Read(2) div 16
            for num = 0 to 255
            if lecture = num then
               if cont[num] < 255 then
                  cont[num]=cont[num]
               end if
             end if
            next num
   end sub

   main:

   ADPCFG = 0xFFF8
   TRISA  = 0xFFFF
   TRISB  = 0x0007
   TRISD  = 0x0000
   TRISF  = 0xFFFF

   for num = 0 to 255
       cont[num] = 0
   next num
   lectura = 0
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice B. **Código fuente práctica conversión 12 bits parte II**

```
ADC1_Init_Advanced(_ADC_INTERNAL_VREFL  or _ADC_EXTERNAL_VREFH)

while (TRUE)
      if PORTF.0 = 1 then
         for num = 0 to 255
         cont[num] = 0
         next num
         while (TRUE)
         leer
             if PORTF.1 = 1 then
             goto dos
             end if
           wend
         end if
      dos:

      for mun = 0 to 255
      PORTB = ((cont[num-3]+cont[num-2]
               +cont[num-1]+cont[num]
               +cont[num+1]+cont[num+2]
               +cont[num+3])div 7)<< 3
      Delay_us(1)
      next num

  wend


  end.
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice C. **Código fuente práctica Transformada de Fourier parte I**

```
program Fourier

dim Samples as word[256] absolute $0C00
freq as word
txt as string[5]
Written as word[64]


sub procedure InitAdc 'Inicialización del conversor AD

ADPCFG = 0x00FF ' PORTB es la entrada análoga
ADCHS = 8 ' Conectar RBxx/Anxx como entrada CH8. RB8 es el pin de la entrada
ADCSSL = 0
ADCON3 = $1F3F ' sample time = 31 Tad
ADCON2 = 0
ADCON1 = $83E0 ' activar el ADC
TRISB.8 = 1 ' RB8 como pin de entrada AD
end sub

'Inicializar el GLCD para el programador EasydsPIC4
sub procedure InitGlcd

    Glcd_Init_EasydsPIC4()
    Glcd_Set_Font(@FontSystem5x8, 5, 8, 32)
    Glcd_Fill(0xAA) ' Show stripes on GLCD to signalize startup
    Delay_ms(500) ' Retardo
    Glcd_Fill(0x00) ' Borrar la pantalla
    end sub

'Función auxiliar para convertir 1.15 en punto base de tipo float (necesita para sacar la raizcuadrada).

sub function Fract2Float(dim input_ as integer) as float

    if (input_ < 0) then
    input_ = - input_
    end if
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice D. **Código fuente práctica Transformada de Fourier parte II**

```
   result = input_/32768.

end sub

' Datos de salida de la sub rutina . estos datos de la sub rutina serán dibujados en el GLCD.
'GLCD coordina el sistema y empieza en la esquina superior izquierda, Por tanto
'el dibujo de la línea tuvo que ser modificada a fin de lograr
' un espectro visible en la pantalla
'Muestra en ese momento contiene DFT de la señal en la manera Re, Im, Re, Im..


sub procedure WriteData

    dim Re, Im, tmpw,
     j, k, l, max as word
     Rer, Imr, tmpR as float

     j = 0 ' Si desea omitir la componente DC luego hacer j> = 1
     k = 0
     max = 0
     freq = 0
     while k <= 63
     Re = Samples[j]
     inc(j)
     Im = Samples[j]
     inc(j)
     Rer = Fract2Float(Re) ' convertir a IEEE punto flotante
     Imr = Fract2Float(Im) ' convertir a IEEE punto flotante
     tmpR = Rer * Rer ' Re^2
     Rer = tmpR
     tmpR = Imr * Imr ' Im^2
     Imr = tmpR
     tmpR = sqrt(Rer + Imr) ' Amplitud de la corriente de la muestra de la TF
     Rer = tmpR*256. ' DFT is scaled down by 1/N, we need to

     ' tomarlo de Nuevo a fin de tener componentes visibles
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice E. **Código fuente práctica Transformada de Fourier parte III**

```
  ' en el GLCD

  Re = Rer
  if Re > 63 then
     if k = 0 then
     Re = 0

     else

     Re = Written[k-1] ' k = 0? tener cuidado con los saltos
     end if

  end if

  if Re > max then ' Encuentra la maxima amplitud

  max = Re

  freq = k ' Esta debe ser la frecuencia central de la seña

 end if



tmpw = Written[k]
if tmpw <> Re then ' Dibuja solo las componentes que son cambiadas
l = 64 - tmpw ' 64 líneas en el GLCD en el eje Y
while l <= 63 ' Limpiar la línea del fondo de la pantalla
Glcd_Dot(k, l, 0)
inc(l)
wend

l = 64 - Re ' dibujar la línea del fondo de la pantalla
while l <= 63
```

Fuente: elaboración propia, *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice F. **Código fuente práctica Transformada de Fourier parte IV**

```
    Glcd_Dot(k, l, 1)
    inc(l)
    wend

    Written[k] = Re ' Marca que la muestra de la corriente ha sido dibujada

    end if

    inc(k) 'Mueve la corriente la coordenada
    inc(k) ' Dibuja en cada segundo la coordenada

    wend

    ' Escribe la frecuencia máxima de la muestra


    freq = freq * 100
    WordToStr(freq, txt)
    Glcd_Write_Text(txt, 70, 0, 1)
end sub



    ' Toma la muestra de corriente



sub function ReadAdc as word

    ADCON1.1 = 1 ' Inicia el conversor AD
             while ADCON1.0 = 0 ' Espera hasta que termine el conversor AD
              nop
              wend
    result = ADCBUF0 ' Obtiene el valor del ADC
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice G. **Código fuente práctica Transformada de Fourier parte V**

```
'llena las muestras con muestras de entrada in la manera Re, Im, Re, Im... donde Im = 0

sub procedure SampleInput

dim i as integer

i =0

while i <= 255

Samples[i] = ReadAdc ' Re

inc(i)

Samples[i] = 0

inc(i) ' Im

wend

' "Muestra " ahora contiene 128 pares de <Re, Im>

end sub

' Main programa inicia aqui

 main:

      'Maininit ' Iniciar todo

      while true ' Lazo infinito
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice H. **Código fuente práctica Transformada de Fourier parte VI**

```
    SampleInput ' Muestra de la señal de entrada


    ' Realizar FFT (DFT), 7 etapas, 128 muestras de pares de complejos

    'Factores Twiddle son tomados de el <TwiddleFactors.dpas>
    FFT(7, @TwiddleCoeff_128, Samples)

    ' DFT mariposa algoritmo de bits de salida se invierte muestras.
    ' Tenemos que restaurar en orden natural.

    BitReverseComplex(7, Samples)

    ' Dibuja la TF en el GLCD

    WriteData

  wend
end.
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice I. **Código fuente práctica PWM**

```
program PWM_Test
dim pwm_period1, pwm_period2, i1, i2 as word

main:

  i1 = 0
  i2 = 0

  pwm_period1 = Pwm_Init(5000, 1, 1, 2)
  Pwm_Init(5000, 3, 1, 2)
  pwm_period2 = Pwm_Init(10000, 2, 1, 3)
  Pwm_Init(10000, 4, 1, 3)

  Pwm_Start(1)
  Pwm_Start(2)
  Pwm_Start(3)
  Pwm_Start(4)

  Pwm_Set_Duty(pwm_period1 div 2, 1)
  Pwm_Set_Duty(pwm_period2 div 2, 2)

  while true
    Pwm_Set_Duty(i1, 3)
    Pwm_Set_Duty(i2, 4)
    if (i1 = pwm_period1) then
      i1 = 0
    end if
    if (i2 = pwm_period2) then
      i2 = 0
    end if
    Inc(i1)
    Inc(i2)
    Delay_ms(1)
  wend
end.
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice J. **Código fuente práctica Frecuencímetro parte I**

```
char cont;
char dece;
char inte;
char pulso,

void InitTumer1(){

T1CON           =0X8000;
TIIE_BIT        =1;
T1IF_BIT        =0;
IPCO            =IPCO | 0X1000;
PR1             =62500;

}

Void Timer1Interupt() iv IVT_ADDR_T1INTERRUPT{
if(T1IF_bit==1)
{
        if(RD1_bit==0)
        {
                if(cont<=9)
                {
                cont++;

                }
                        if(cont==9)
                        {
                        cont=0
                        dece++
                        }
                        if(cont==9)&&(dece==9)
                                {
                                cont=0
                                dece=0

                                }
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice K. **Código fuente práctica Frecuencímetro parte II**

```
                                        }


                                        do
                                        {

                                        }while(RD1_bit==0);
                        }
                        inte++
                        T1IF_bit=0;
                }
}


void main()
{
        ADPCFG=0XFFFF;
        TRISB=0
        TRISF=0
        TRISD=0XFFFF;
        CONT=0;
        Inte=0;
        pulso=0;
        dece=0;
        decepulso=0;
        do
        {
                if(RD0_bit==0)
                {
                InitTimer1()
                        if(inte==40)
                        {
                        LATF=cont;
                        LATB=dece;
                        delay_ms(500);
                        inte=0;
```

Fuente: Elaboración Propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice L. **Código fuente práctica Frecuencímetro parte III**

```
        cont=0;
        dece=0;
        LATF=cont;
        LATB=dece;

        PR1             =62500;

        }
}
else
{
if(RD0_bit==1)
        {
                if(!RD2_bit)
                        {
                                if(pulso<=9)
                                {
                                        pulso++;
                                        LAFT=pulso;
                                        if((pulso==9)&&(decepulso==9))
                                {
                                                pulso=0;
                                                decepulso=0;
                                                LATB=decepulso;
                                                LATF=pulso;

                                        }
                        }
                        if(pulso=9)
                        {
                                pulso=0;
                                decepulso++;
                                LATB=decepulso;
                                LATF=pulso;

                        }
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice M. **Código fuente práctica Frecuencímetro parte IV**

```
                                        if(pulso=9)
                                        {
                                                pulso=0;
                                                decepulso++;
                                                LATB=decepulso;
                                                LATF=pulso;

                                        }
                                        do
                                        {
                                        }while(!RD2_bit);
                                        delay_ms(50);
                                        }
                                }
                        }

        }while(1);
}
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice N. **Código fuente práctica RS-232 parte I**

```
Include "P30f4013.inc"
global_U1RXInterrupt

_U1RXInterrupt:
BCLR    IFS0,#U1RX1F;
MOV     U1RXREG,W7;
MPY     W7*W7,A
MOV     ACCAL,W2
MOV     #0x0064,W3
REPEAT  #17
DIV.U   W2,W3
CALL    TRANSMITE
MOV     W1,W0
CALL    TRANSMITE
RETFIE


.global_main

_main:

BSET     CORCON,#0x0
CALL     INICIAUART
bucle:
CLRWDT
GOTO     bucle
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.0.

Apéndice O. **Código fuente práctica RS-232 parte II**

```
INICIAUART:
CLR
MOV      #0X0019,WO
MOV      W0,U1BRG
MOV      #0X800,WO
MOV      WO,U1MODE
MOV      #0x0510,WO
MOV      WO,1USTA
MOV      #0x8020,WO
MOV      WO,U1MODE
MOV      #0x0200,WO
MOV      WO,ieCO
CLR      IEC1
CLR      IEC2
RETURN


TRANSMITE:

BTSS     U1STA,#8
BRA      TRANSMITE
MOV      WO,U1TXREG
RETURN
.end
```

Fuente: elaboración propia, con programa *mikroBasic compiler for dsPIC* v 5.0.0.