IMPLEMENTACIÓN DE FILTROS DIGITALES UTILIZANDO EL PC

JORGE ELIÉCER LOZADA SERRANO
OSCAR ALBERTO RINCÓN MARTÍNEZ
CÉSAR ARMANDO TOVAR CORREA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO – MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA, MARZO DE 2005

IMPLEMENTACIÓN DE FILTROS DIGITALES UTILIZANDO EL PC

JORGE ELIÉCER LOZADA SERRANO
OSCAR ALBERTO RINCÓN MARTÍNEZ
CÉSAR ARMANDO TOVAR CORREA

Proyecto de grado presentado como requisito para optar al título de ingeniero electrónico

Director
Mpe. JAIME BARRERO PÉREZ
Codirector
Ing. JAVIER GONZÁLEZ BARAJAS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO – MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA, MARZO DE 2005

A Dios por acompañarme siempre
A mi padre por ser mi ángel de la guarda
A mi madre y a Helffar por su inmensa confianza y apoyo
A mis hermanos por su inacabable comprensión y paciencia

Jorge Eliécer Lozada

A mis padres, por la paciencia, confianza y apoyo
incondicional,
A mi hermana por ayudarme en los
momentos más difíciles,
A Dios por su omnipresencia
en todo momento

César Armando Tovar

A Dios y mi familia

Oscar Alberto Rincón

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

Jaime Barrero, magister en potencia eléctrica y director del proyecto, por su valiosa orientación y su colaboración. Por la confianza depositada en nosotros, y la motivación de trabajar en el campo del filtrado digital.

Javier González, ingeniero electrónico y codirector del proyecto, por su ayuda y dedicación durante todo el proyecto.

Y en general a todas las personas que sirvieron de apoyo para el desarrollo de este proyecto, por su amable disposición y asesoría cuando fue necesaria.

TABLA DE CONTENIDO

Pag.

LISTA DE FIGURAS

Pag.

LISTA DE TABLAS

LISTA DE ANEXOS

TÍTULO: IMPLEMENTACIÓN DE FILTROS DIGITALES UTILIZANDO EL PC[*]

AUTORES: Jorge Eliécer Lozada Serrano
Oscar Alberto Rincón Martínez
César Armando Tovar Correa[**]

PALABRAS CLAVES: Filtros Digitales, Filtros IIR, Filtros FIR, Generador de Señales Arbitrarias, Tarjeta de Adquisición de Datos PCI 1200, LabVIEW, Filtrado en Tiempo Real, Técnica de Enventaneo, Transformada Bilineal, Técnica de Rizado Constante.

DESCRIPCIÓN:

El propósito de este proyecto de grado es implementar filtros digitales en un computador personal para realizar el filtrado en tiempo real de una gran variedad de señales analógicas. El filtrado en tiempo real se logró utilizando un sistema de procesamiento digital basado en la tarjeta de adquisicón de datos PCI 1200 de National Instruments, el software de aplicación LabVIEW 6.i y un computador personal. Las señales analógicas se obtuvieron de un generador de señales arbitrarias programado en LabVIEW el cual genera señales comunes (señales senoidales, triangulares, sierras y cuadradas), señales de fórmulas matemáticas, señales dibujadas con el ratón del PC y señales importadas de archivos de texto. El filtrado digital se implementó en el lenguaje de programación gráfico de LabVIEW en una estructura de bloques. El primer bloque se encarga de la adquisición de la señal analógica. En el segundo bloque el usuario puede diseñar filtros digitales IIR o FIR. Los filtros IIR pueden diseñarse usandd las topologías de Butterworth, Chebyshev, Chebyshev inverso, Bessel y elípticas. Los filtros FIR pueden diseñarse con la técnica de enventaneo (incluyendo, entre otras, ventana rectangular, Hamming, Hannig y Kaiser) y la técnica de rizado constante que utiliza el algoritmo de Parks-McClellan Se pueden diseñar con esta aplicación filtros pasa bajas, pasa altas, pasa banda y rechaza banda; especificando parámetros de diseño como atenuaciones en la banda de paso y rechazo, frecuencias de corte, orden y máximo rizado permitido. Ademas se pueden diseñar filtros introduciendo los coeficientes de la función de transferencia *H(z).* La aplicación de diseño visualiza la magnitud de la función de transferencia del filtro diseñado, en escala lineal o en decibeles; y también visualiza la respuesta de fase del filtro en radianes o en grados, en escala continua o discontinua. El tercer bloque de procesamiento filtra en tiempo real la señal adquirida y visualiza la señal resultante y su espectro de potencia en el PC luego de ser procesada por el filtro diseñado.

---

[*] Proyecto de grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director del proyecto, Mpe. Jaime Barrero Pérez.

TITLE: DIGITAL FILTER IMPLEMENTATION USING PC*

AUTORS: Jorge Eliécer Lozada Serrano
Oscar Alberto Rincón Martínez
César Armando Tovar Correa**

KEY WORDS: Digital Filters, IIR Filters, FIR Filters, Arbitrary Waveform Generator, PCI 1200 Data Acquisition Board, LabVIEW, Real Time Filtering, Windowing Design Technique, Bilinear transformation, Equi-Ripple Design Technique

DESCRIPTION:

The purpose of this project is the digital filter implementation using a personal computer to perform real time filtering on a wide variety of analogic signals. Real time filtering was achieved using a digital processing system based on a PCI 1200 data acquisition board, LabVIEW application software and a personal computer. Analogic signals were obtained by means of a LabVIEW software created arbitrary waveform generator wich generates common signals (sine, triangular, sawthoot and square signals), mathematical formulated signals, PC mouse sketched signals and text file imported signals. Digital filtering was implemented using LabVIEW graphical programming lenguage in a block structure. The first block deals with analogic signal acquisition. In the second block the user can design FIR or IIR digital filters. IIR filters can be designed using Butterworth, Chebychev, inverse Chebyshev, Bessel and eliptic topolygies. FIR filter can be designed with windowing technique (including, among others, rectangular, Hamming, Hannig and Kaiser window) and equi-ripple technique wich uses the Parks-McClellan algoritm. Low pass, high pass, band pass and band reject digital filters can be design with this software application, specifying design parameters such as pass and stop band attenuations, cut-off frecuencies, order, and maximun ripple allowed. Moreover, filters can be designed introducing the function tranfers *H(z)* coefficients. The design application displays function transfer magnitude in linear or decibel scale; also it displays the filter phase respone in degree or radians, in continue or warp scale. The digital processing third block filters in real time the acquire signal and displays its power spectrum and the resultant signal in the PC after it was processed by the filter designed.

---

* Degree Project

** Faculty of Physical-Mechanical Sciences. Electric, Electronic and Telecommunication Engineering School. Project Director, Mpe. Jaime Barrero Pérez.

UNIVERSIDAD INDUSTRIAL DE SANTANDER

NOTA DEL PROYECTO DE GRADO

Nombre del Estudiante: JORGE ELIÉCER LOZADA SERRANO		CÓDIGO: 1982645
TITULO DEL PROYECTO: IMPLEMENTACIÓN DE FILTROS DIGITALES EN EL PC		
CÓDIGO: 01200410	FACULTAD: INGENIERÍAS FISICOMECÁNICAS	CARRERA: INGENIERÍA ELECTRÓNICA
CALIFICACIÓN: CUATRO, CINCO (4.5)		CRÉDITOS: 15

DIRECTOR DEL PROYECTO

NOMBRE: JAIME GUILLERMO BARRERO PÉREZ	FIRMA

CALIFICADORES

F N CÉSAR ANTONIO DUARTE GUALDRÓN	F Nidia Quintero Peña N NIDIA QUINTERO PEÑA	FECHA: A 2005 | M 04 | D 15

UNIVERSIDAD INDUSTRIAL DE SANTANDER

NOTA DEL PROYECTO DE GRADO

Nombre del Estudiante: OSCAR ALBERTO RINCÓN MARTÍNEZ		CÓDIGO: 1982216
TITULO DEL PROYECTO: IMPLEMENTACIÓN DE FILTROS DIGITALES EN EL PC		
CÓDIGO: 01200410	FACULTAD: INGENIERÍAS FISICOMECÁNICAS	CARRERA: INGENIERÍA ELECTRÓNICA
CALIFICACIÓN: CUATRO, CINCO (4.5)		CRÉDITOS: 15

DIRECTOR DEL PROYECTO

NOMBRE: JAIME GUILLERMO BARRERO PÉREZ	FIRMA

CALIFICADORES

F N CÉSAR ANTONIO DUARTE GUALDRÓN	F Nidia Quintero Peña N NIDIA QUINTERO PEÑA	FECHA: A 2005 | M 04 | D 15

UNIVERSIDAD INDUSTRIAL DE SANTANDER

NOTA DEL PROYECTO DE GRADO

Nombre del Estudiante: CÉSAR ARMANDO TOVAR CORREA		CÓDIGO: 1982273
TITULO DEL PROYECTO: IMPLEMENTACIÓN DE FILTROS DIGITALES EN EL PC		
CÓDIGO: 01200410	FACULTAD: INGENIERÍAS FISICOMECÁNICAS	CARRERA: INGENIERÍA ELECTRÓNICA
CALIFICACIÓN: CUATRO, CINCO (4.5)		CRÉDITOS: 15

DIRECTOR DEL PROYECTO

NOMBRE: JAIME GUILLERMO BARRERO PÉREZ	FIRMA

CALIFICADORES

F N CÉSAR ANTONIO DUARTE GUALDRÓN	F Nidia Quintero Peña N NIDIA QUINTERO PEÑA	FECHA: A 2005 | M 04 | D 15

INTRODUCCIÓN

El avance en las investigaciones de la teoría digital y las nuevas tecnologías desarrolladas en las últimas décadas en el terreno de la microelectrónica, han hecho del procesamiento digital de las señales un campo que encuentra aplicaciones cada vez más sofisticadas que necesitan desempeños superiores a los que pueden ofrecer los sistemas analógicos.

Ya en la década de los sesenta, el tratamiento de señales en tiempo discreto hacía sus primeras apariciones, provocando diversas reacciones entre los defensores del procesamiento analógico. En ese momento los computadores disponibles eran posesiones privilegiadas de grandes industrias, universidades o laboratorios científicos debido a su alto costo y gran consumo de potencia. Estos equipos tenían una capacidad computacional limitada lo cual restringía su uso en aplicaciones que necesitaran procesamiento en tiempo real, ya que era algo común que se necesitaran minutos o incluso horas de tiempo de computador para procesar sólo algunos segundos de datos. A pesar de estas deficiencias, los computadores se manifestaban como una forma atractiva de simular sistemas analógicos para ser estudiados en entornos experimentales virtuales y flexibles antes de comprometer los recursos económicos necesarios para ensamblar el hardware analógico.

Estas simulaciones fueron la base de los primeros trabajos que se realizaron en filtrado digital. Se estudiaron entonces las formas para que una señal analógica que fuera digitalizada con un conversor A/D se pudiera procesar mediante un programa en un computador de tal manera que luego de realizar una conversión D/A de la señal resultante, el sistema completo se comportará aproximadamente como un buen filtro analógico. Sin embargo, las expectativas de que estos sistemas pudieran llegar a ser prácticos en el procesamiento en tiempo real de señales eran poco optimistas debido a

tres restricciones principales, sobre las cuales tenían una gran ventaja los sistemas analógicos en ese momento: la velocidad, el tamaño y el costo.

Con el desarrollo de nuevos algoritmos matemáticos[3] que redujeron el tiempo de cómputo discreto, se logró realizar el tratamiento digital de señales con programas cada vez más complejos y con un tiempo de respuesta que permitía una interacción en tiempo real. Fue así como la literatura fue abordando con más detalle el tema de los sistemas en tiempo discreto, reformulando una serie de propiedades y conceptos matemáticos para trabajar con secuencias de muestras, que no eran aproximaciones a sistemas continuos si no un campo matemático totalmente independiente, que tenía relaciones y fórmulas exactas en tiempo discreto. Esta nueva onda en que los sistemas de tratamiento de señales en un computador no eran una simple aproximación de técnicas analógicas despertó un interés todavía mayor ya que el tratamiento de señales en tiempo discreto merecía ser investigado de forma independiente. Para mediados de los años ochenta con la llegada de los microprocesadores de bajo costo, producción en masa, bajo consumo de potencia y gran velocidad, fue posible la implementación de sistemas en tiempo discreto para el procesamiento de una señal en tiempo real en una amplia gama de aplicaciones.

Este proyecto tomó provecho de la evolución de los sistemas digitales y de la microelectrónica para implementar un sistema de procesamiento en tiempo discreto de una señal, más específicamente el procesamiento de una señal filtrada digitalmente. Se transformó un computador personal en una herramienta para el filtrado digital de una señal analógica, de forma tal que se logró visualizar en tiempo real en el PC la señal de entrada y de salida del filtro digital.

Esta idea surgió gracias a la investigación respecto al tratamiento de señales electrocardiográficas que se lleva a cabo en la Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones con el fin de avanzar en este campo e impulsar el desarrollo de los filtros digitales y el procesamiento digitales de señales en la escuela. El problema con

[3] Como el algoritmo de la FFT (FFT, *Fast Fourier Transform*) descubierto por Cooley y Tukey (1965), que disminuyó el número de operaciones necesarias para calcular las transformadas de Fourier, lo cual redujo significativamente el tiempo de proceso de los computadores.

la señal electrocardiográfica es el hecho de que es una suma de impulsos eléctricos de muchos músculos, no solamente el corazón; además de otros ruidos como es el ruido de 60Hz y el ruido por inducción electromagnética. Por tanto para visualizar en una pantalla un ECG libre de ruido se necesitan filtros de alto orden, gran flexibilidad y baja inmunidad al ruido, y estas características son la parte llamativa de los filtros digitales.

El objetivo del proyecto fue implementar filtros digitales en LabVIEW y evaluar su desempeño en tiempo real en un PC. Para lograr esto fue necesario primero recopilar y asimilar información sobre la forma de programación de LabVIEW y como controlar la tarjeta de adquisición de datos PCI 1200 de National Instruments desde este software. También se recopiló información sobre el tratamiento de señales en tiempo discreto, haciendo énfasis en las técnicas de diseño de filtros digitales, con ejemplos detallados y algunos ejemplos de diseño de filtros digitales en MATLAB.

En el capítulo uno se presenta una recopilación teórica del filtrado digital de señales analógicas, desde el proceso de muestreo hasta el tratamiento digital a través de filtros recursivos y no recursivos, explicando las herramientas matemáticas disponibles para facilitar la manipulación de las señales en tiempo y frecuencia. Se explican con cierto detalle las técnicas de diseño más comúnmente utilizadas y que se pueden programar dentro de la aplicación de filtrado digital en tiempo real implementada en LabVIEW. Dentro de los filtros IIR (recursivos) se examina el diseño mediante transformada bilineal; mientras que en los filtros FIR (no recursivos), se detallan los métodos de diseño de filtros con ventanas (incluyendo ventana rectangular, triangular, Hanning, Hamming, Kaiser-Bessel y Blackman) y algoritmos que minimizan el error máximo de la magnitud de la respuesta en frecuencia del filtro como el diseño óptimo con rizado constante.

El capítulo dos se dedica a la explicación del algoritmo implementado en LabVIEW para generar señales eléctricas de diversos tipos, creadas a partir de librerías, bases de datos, expresiones matemáticas y señales aleatorias dibujadas con el ratón del computador. Esto con el fin de tener un banco de señales de prueba con espectros conocidos que sirvieran de entrada a la aplicación de filtrado digital y además para tener una herramienta de laboratorio capaz de generar una gran gama de señales arbitrarias útiles en diversas aplicaciones.

El capítulo tres se enfoca en el desarrollo de una aplicación programada en LabVIEW capaz de adquirir una señal eléctrica por medio de una tarjeta de adquisición de datos y procesarla en tiempo real a través de filtros digitales FIR e IIR implementados mediante instrumentos virtuales y una interfaz gráfica que permite observar la señal de salida y su espectro en frecuencia luego de ser procesada.

La realización de pruebas al generador de señales y a los filtros digitales desarrollados se encuentran en el capítulo 4, donde se pueden observar, entre otros, los resultados obtenidos al filtrar señales analógicas con espectros conocidos generadas a partir de la aplicación explicada en el capítulo 2.

1. FILTROS DIGITALES

El tema principal de este proyecto son los filtros digitales, los cuales extraen o modifican alguna característica de la señal de entrada; que se ve reflejada en su espectro de frecuencia, por ejemplo se habla del paso de una determinada banda del espectro de la señal de entrada. Por lo tanto, los filtros digitales se definen como sistemas discretos que modifican características en frecuencia de una señal digital.

Ante esta definición, es necesario comprender conceptos básicos de los sistemas discretos. En el Anexo C se encuentra una base teórica de sistemas discretos.

El diseño de los filtros requiere de las siguientes etapas: (1) especificaciones de las propiedades deseadas del sistema, (2) aproximaciones de las especificaciones mediante un sistema causal en tiempo discreto y (3) la realización del sistema. El primero es altamente dependiente de la aplicación y el tercero de la tecnología utilizada para la implementación.

En la práctica el filtro deseado se realiza utilizando cómputo digital y se emplea para una señal analógica seguida por una conversión analógico-digital.

Cuando se utiliza un filtro digital para realizar el tratamiento de señales en tiempo continuo, tanto las especificaciones del filtro en tiempo discreto como las que se indica en tiempo continuo se suelen dar en el dominio de la frecuencia

Los filtros digitales muestran abrumadoras ventajas respecto a los sistemas analógicos, una enumeración de los beneficios puede verse dentro de la comparación entre estos dos tipos de filtros:

- 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.

- Conmutabilidad: Los parámetros de los filtros digitales pueden ser modificados a voluntad. De esta forma, estos filtros se pueden transformar obteniendo otro filtro, pudiéndose multiplexar en el tiempo para procesar varias señales de entrada.

- Adaptabilidad: Un filtro digital puede ser implementado en soporte físico (hardware) o mediante un programa (software).

- Ausencia de problemas de componentes: Los parámetros de los filtros digitales 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.

- Complejidad: La potencia de cálculo de los computadores actuales y de los algoritmos desarrollados, permiten implementar aplicaciones casi imposibles de diseñar con filtros analógicos.

Una distinción fundamental en los sistemas discretos lineales e invariantes, y en particular en los filtros digitales, es la duración de la respuesta ante el impulso. Se habla de sistemas de respuesta al impulso finito o no recursivo FIR, *(Finite Impulse Response*) y de sistemas de respuesta infinita o recursivo IIR, (*Infinite Impulse Response*). Partiendo de la ecuación en diferencias que modela el comportamiento dinámico de estos sistemas:

$$y[n] = a_1 y[n-1] + a_2 y[n-2] + ... + a_m y[n-m] + b_o x[n] + b_1 x[n-1] + b_2 x[n-2] + ... + b_k x[n-k] \quad (1.1)$$

En el caso de tener todos los coeficientes ($\mathbf{a_i}$) iguales a cero se tendrá un filtro FIR, con lo que quedará la ecuación reducida a:

$$y[n] = b_{ox}x[n] + b_1x[n-1] + b_2x[n-2] + ... + b_kx[n-k] \quad (1.2).$$

Los filtros FIR son aquellos en los cuales la salida depende sólo de valores presentes y pasados de la entrada, Siendo k el orden del filtro y tendrá una función de transferencia en z igual a:

$$H(z) = b_o + b_1z^{-1} + b_2z^{-2} + ... + b_kz^{-k} \qquad (1.3).$$

De las Ecuaciones (1.2) y (1.3) se observa que en estos filtros cada valor de la secuencia de salida sólo depende de un número finito de valores de la secuencia de entrada. Además también se desprende la carencia de polos en la función de transferencia. Por otra parte, las expresiones de los filtros recursivos IIR corresponden a:

$$Y[n] = a_1Y[n-1] + a_2Y[n-2] + ... + a_mY[n-m] + b_oX[n] + b_1X[n-1] + b_2X[n-2] + ... + b_kX[n-k] \quad (1.4)$$

Y su función de transferencia en z es:

$$H(z) = \frac{b_o + b_1z^{-1} + b_2z^{-2} + ... + b_kz^{-k}}{1 - a_1z^{-1} - a_2z^{-2} - ... - a_mz^{-m}} \qquad (1.5).$$

En estos casos, la secuencia de salida depende tanto de la entrada como de la salida. De las Ecuaciones (1.1) a la (1.5) se deducen las siguientes propiedades.

- La secuencia de ponderación de la respuesta al impulso 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.

- Los filtros FIR son siempre estables, esto significa que la secuencia de salida tiene todos sus valores acotados. No es el caso de los filtros recursivos, ya que su estabilidad depende de la función de transferencia, por lo que se deberá utilizar alguno de los procedimientos algebraicos para analizar su estabilidad.

- Cualquier filtro recursivo puede ser reemplazado por otro no recursivo con infinitos coeficientes y sus valores estarán dados por la secuencia de ponderación del IIR. La conclusión inversa no se cumple.

1.1 SISTEMAS DE FASE LINEAL

Cuando se diseñan filtros y otros sistemas que dejan pasar sin distorsión una banda de frecuencias, es deseable que el módulo de la respuesta en frecuencia sea aproximadamente constante y de fase cero en la banda de paso. Para efectos prácticos (sistemas causales) se debe permitir cierta distorsión de fase. Una fase lineal es simplemente un desplazamiento temporal que no afecta la forma de la señal de entrada.

Considérese un sistema LTI con respuesta en frecuencia

$$H(e^{jw}) = e^{-jw\alpha}, \text{ para } |w| < \pi \quad (1.6).$$

Para este sistema se tiene:

$$|H(e^{jw})| = 1 \quad (1.7)$$

$$\text{Fase} = -w\alpha \quad (1.8)$$

La transformada inversa de Fourier de la Ecuación (1.6) es:

$$h[n] = \frac{sen\pi(n-\alpha)}{\pi(n-\alpha)}, \text{ para } -\infty < n < \infty \quad (1.9).$$

Ahora si α es entero entonces h[n] = $\delta[n-\alpha]$ y y[n] = x[n] $*\delta[n-\alpha]$ = x[n-α], es decir si α es un entero entonces el sistema de fase lineal y ganancia uno simplemente desplaza la secuencia de entrada α muestras.

Un sistema de fase lineal se comporta en general como:

$$H(e^{jw}) = |H(e^{jw})| e^{-jw\alpha}, \text{ para } |w| < \pi \quad (1.10).$$

Lo que indica que la salida y[n] de un sistema LTI de fase lineal es la convolución de la señal de entrada x[n] con la magnitud de la respuesta al impulso $|H(e^{jw})|$ del sistema, mas un desplazamiento en tiempo α muestras.

Considérese el caso del filtro pasa bajo ideal, donde

$$H_{LP}(e^{jw}) = e^{-jw\alpha}, \text{ para } |w| < w_C \text{ y cero en el resto } (1.11).$$

Entonces

$$h_{LP}[n] = \frac{sen w_c(n-\alpha)}{\pi(n-\alpha)} \quad (1.12).$$

De la Ecuación (1.12) se concluye que si 2α es un numero entero entonces $h_{LP}[n]$ es simétrica respecto a α.

Para que el sistema sea de fase lineal generalizada es necesario que su respuesta al impulso sea simétrica o antisimétrica respecto a α. Si el sistema es causal es decir h[n] es cero para n < 0 significa que la respuesta al impulso debe truncarse a un valor M = 2α lo que da como resultado:

$$h[n] = h[M-n]$$
$$\text{Para } 0 \le n \le M \text{ y cero en el resto} \quad (1.13).$$

$$h[n] = - h[M-n]$$

Este es el caso de los filtros FIR que son causales y de fase lineal generalizada.

1.2 FILTROS FIR SIMÉTRICOS Y ANTISIMÉTRICOS

Un filtro FIR tiene fase lineal si su respuesta al impulso satisface la condición de simetría o antisimetría de sus coeficientes. Para su demostración se partirá de la respuesta en frecuencia de un filtro no recursivo que tenga un orden M. Además, para facilitar su comprensión se va a suponer que el filtro sea de orden par, definiéndose que M sea igual a 2N, por tanto, la respuesta en frecuencia de los filtros de orden par quedará como:

$$H\left(e^{jw}\right) = \sum_{n=0}^{2N} h[n] e^{-jwn} = e^{-jwN}\{h[0]e^{jwN}+h[1]e^{j(N-1)w}+h[2]e^{j(N-2)w}+\ldots+h[2N]e^{-jwN}\} \quad (1.14).$$

Para simetría o antisimetría se tiene:

$$h[0] = \pm h[2N]$$
$$h[1] = \pm h[2N-1]$$
$$h[2] = \pm h[2N-2]$$
$$.$$
$$.$$
$$.$$
$$h[N-1] = \pm h[N+1]$$

El signo (+) indica simetría en los coeficientes y el (–) para la antisimetría.

Suponiendo primero que existe simetría de orden par y agrupando alrededor de los coeficientes $e^{\pm j(N-i)w}$ en la Ecuación (1.14), la respuesta en frecuencia del filtro será:

$$H\left(e^{jw}\right) = e^{-jwN}\left(h[N] + 2\sum_{i=0}^{N-1} h[i]\cos[(N-i)w]\right) \quad (1.15).$$

De la Ecuación (1.15) se observa que el contenido que está entre los paréntesis es real y que el desfase introducido por el filtro es -wN, siendo por tanto el argumento, lineal con la frecuencia.

De igual manera se procede con los coeficientes antisimétricos, sin embargo, hay que destacar que si el filtro es de orden par el punto central de la antisimetría será nulo. Es fácil demostrar que la respuesta en frecuencia del un filtro FIR de orden par con respuesta al impulso antisimétrica se puede expresar como:

$$H\left(e^{jw}\right) = e^{-j\left(wN-\frac{\pi}{2}\right)}\left(2\sum_{i=0}^{N-1} h[i]sen[(N-i)w]\right) \quad (1.16).$$

De la Ecuación (1.16) se observa que la característica de fase del filtro es $\frac{\pi}{2} - wN$. Igualmente las expresiones de las respuestas en frecuencia para filtros de orden impar con simetría o antisimetría son sencillas de obtener.

$$H\left(e^{jw}\right) = e^{-jwN}\left(2\sum_{i=0}^{((M+1)/2)-1} h[i]\cos[(M/2-i)w]\right) \quad \text{Simetría de orden impar. (1.17).}$$

$$H\left(e^{jw}\right) = e^{-j\left(wN-\frac{\pi}{2}\right)}\left(2\sum_{i=0}^{((M+1)/2)-1} h[i]sen[(M/2-i)w]\right) \quad \text{Antisimetría de orden impar. (1.18).}$$

Las Ecuaciones (1.15) a la (1.18) son formulas de respuesta en frecuencia generales que se pueden usar para diseñar filtros FIR de fase lineal con respuesta al impulso simétricas y antisimétricas. Nótese que para un filtro simétrico, el número de coeficientes que definen la respuesta es M/2 cuando M es par ó ((M+1)/2)-1) si M es impar. Por otro lado, si la

respuesta al impulso es antisimétrica y el orden es par, el punto central de la antisimetría será nulo y habrá M/2 coeficientes que lo definan, en caso de ser orden impar, cada coeficiente tiene un término de igual magnitud y de signo opuesto y estará definido por ((M+1)/2)-1). La elección de una respuesta al impulso simétrica o antisimétrica depende de la aplicación. Por ejemplo, si el filtro tiene antisimetría, $H(e^{jw})$ tanto para bajas frecuencias como para frecuencias alrededor de la frecuencia de Nyquist tiene valores próximos a cero, por lo que no es posible utilizarlos ni para filtros pasa bajas ni paso alto. Por otro lado, la condición de simetría produce un filtro FIR de fase lineal con una respuesta distinta de cero para bajas frecuencias. En resumen, el problema de diseño de filtros FIR es simplemente el del determinar M+1 coeficientes, a partir de las especificaciones en las frecuencias deseadas.

La solución en Matlab de la respuesta al impulso simétrica de orden par con M = 4 (tipo I) se ilustra en la Figura 1.1.

```
» x=(0:1:4);
» y=ones(1,5);
» stem(x,y)
```

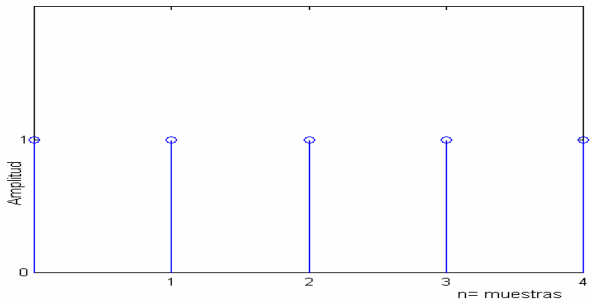


Figura 1.1. Secuencia Tipo (I)

Solución en Matlab de la repuesta al impulso simétrica de orden impar con M = 3 (tipo II) se ilustra en la Figura 1.2.

```
» y=ones(1,4);
» stem(x,y)
```

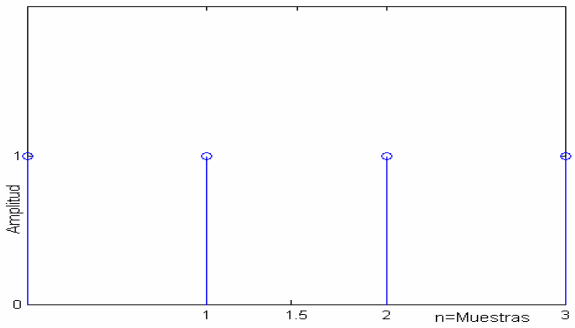


Figura 1.2. Secuencia Tipo (II)

La solución en Matlab de la repuesta al impulso antisimétrica de orden par con M = 4 (tipo III) se ilustra en la Figura 1.3

```
» x=(0:4);
» y=[1,1,0,-1 -1];
» stem(x,y)
```

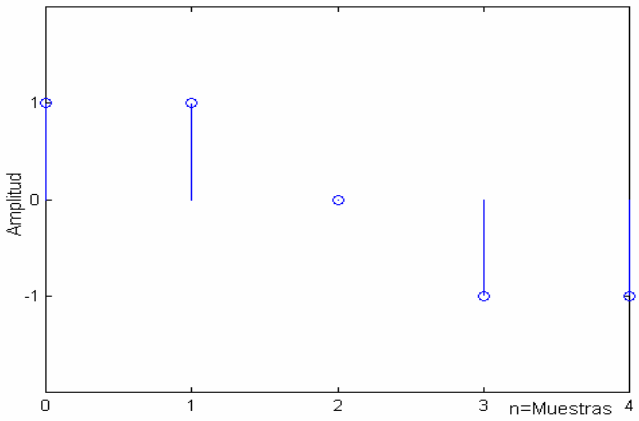


Figura 1.3 Secuencia Tipo (III)

De la Figura 1.3 se observa que para los filtros tipo (III) el punto de simetría M/2 es nulo.

La solución en Matlab de la repuesta al impulso antisimétrica de orden impar con M = 3 (tipo IV) se ilustra en la Figura 1.4.

```
» x=(0:3);
```

» y=[1 1 -1 -1];

» stem(x,y)

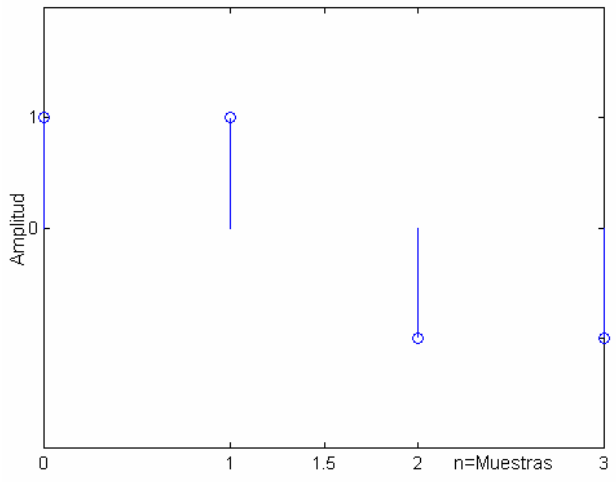


Figura 1.4. Secuencia Tipo (IV)

Tabla 1.1 Secuencias simétricas

TIPO	SIMETRÍA	ORDEN	|H(0)|	|H(π)|
I	PAR	PAR	h[M/2] + $2\sum_{i=0}^{N-1} h[i]$	H[M/2]+ $2\sum_{i=0}^{N-i}(-1)^i h[i]$
II	PAR	IMPAR	$2\sum_{i=0}^{((M+1)/2)-1} h[i]$	0
III	IMPAR	PAR	0	0
IV	IMPAR	IMPAR	0	$2\sum_{i=0}^{((M+1)/2)-1}(-1)^i h[i]$

Tabla 1.2 Aplicaciones a Tipos de Filtros

Aplicaciones de las Secuencias Simétricas		
Tipo	**Observaciones a |H(e^{jw})|**	**Aplicación**
1		Todo tipo de filtros
2	|H(π)=0|	LP, BP y BS
3	|H(0)|=0=|H(π)|	Sólo BP y BS
4	|H(0)|=0	HP, BP y BS

LP: Filtro Pasa Bajas BP:Filtro Pasa Banda
HP:Filtro Pasa Altas BS:Filtro Para Banda

De acuerdo a las Tablas 1.1 y 1.2 se determina como es la secuencia para cada tipo de filtro FIR en general.

1.3 DISEÑO DE 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 frecuencias de fase lineal. Los FIR son por su propia constitución estables. Aunque el diseño de los FIR requiera de una gran cantidad de operaciones de sumas y multiplicaciones, tanto su estructura de programación como su realización en soporte físico resultan fácil y escalable.

En la práctica, los filtros FIR se emplean en problemas de filtrado donde hay un requisito de fase lineal dentro de la banda de paso del filtro. Si no existe este requisito se pueden emplear tanto filtros FIR como IIR. Sin embargo, como regla general, un filtro IIR tiene menos rizado y el corte es más abrupto que un FIR con el mismo grado de polinomio. Por esta razón, si se puede tolerar alguna distorsión de fase o ésta no es importante, se prefiere un IIR, principalmente porque su implementación involucra menos parámetros, requiere menos memoria y tiene menor complejidad computacional.

1.3.1 Diseño de Filtros FIR de Fase Lineal Usando Ventanas

Las técnicas de diseños de filtros FIR se basan en aproximar directamente la respuesta en frecuencia deseada del sistema en tiempo discreto.

El método más sencillo de diseño de filtros FIR se llama método de ventanas, este método empieza definiendo una respuesta ideal deseada.

$$H_d(e^{jw}) = \sum_{n=-\infty}^{\infty} h_d[n] e^{-jwn} \quad (1.19).$$

La secuencia de respuesta al impulso deseada $h_d[n]$ es no causal de longitud infinita, la forma directa de obtener una aproximación FIR causal es truncar la respuesta al impulso y se redefine así.

$$h[n] = h_d[n], \text{ para } 0 \leq n \leq M \text{ y cero en el resto. (1.20).}$$

O de forma más general se puede expresar

$$h[n] = h_d[n]w[n] \quad (1.21)$$

Donde w[n] se define:

$$w[n] = 1, \text{ para } 0 \leq n \leq M \text{ y cero en el resto. (1.22).}$$

De las Ecuaciones (1.21) y (1.22) puede observarse la forma como la respuesta de un filtro FIR tiende a la forma ideal en la medida en que se aumenta el número de coeficientes como se indica en las Figuras (1.5). Ello se manifiesta en que las regiones de transición y el rizado se hacen menos notables a medida que aumenta el número de coeficientes. Si se utiliza un número infinito de coeficientes, el filtro aproximado que se diseña es perfecto. Esto permite concluir que los defectos en la respuesta en frecuencia no nacen debido al método aplicado, sino que surgen por utilizarse una cantidad limitada de coeficientes.

Limitar o truncar el número de coeficientes es necesario por razones prácticas, ya que no es posible implementar un filtro con un número infinito de coeficientes.

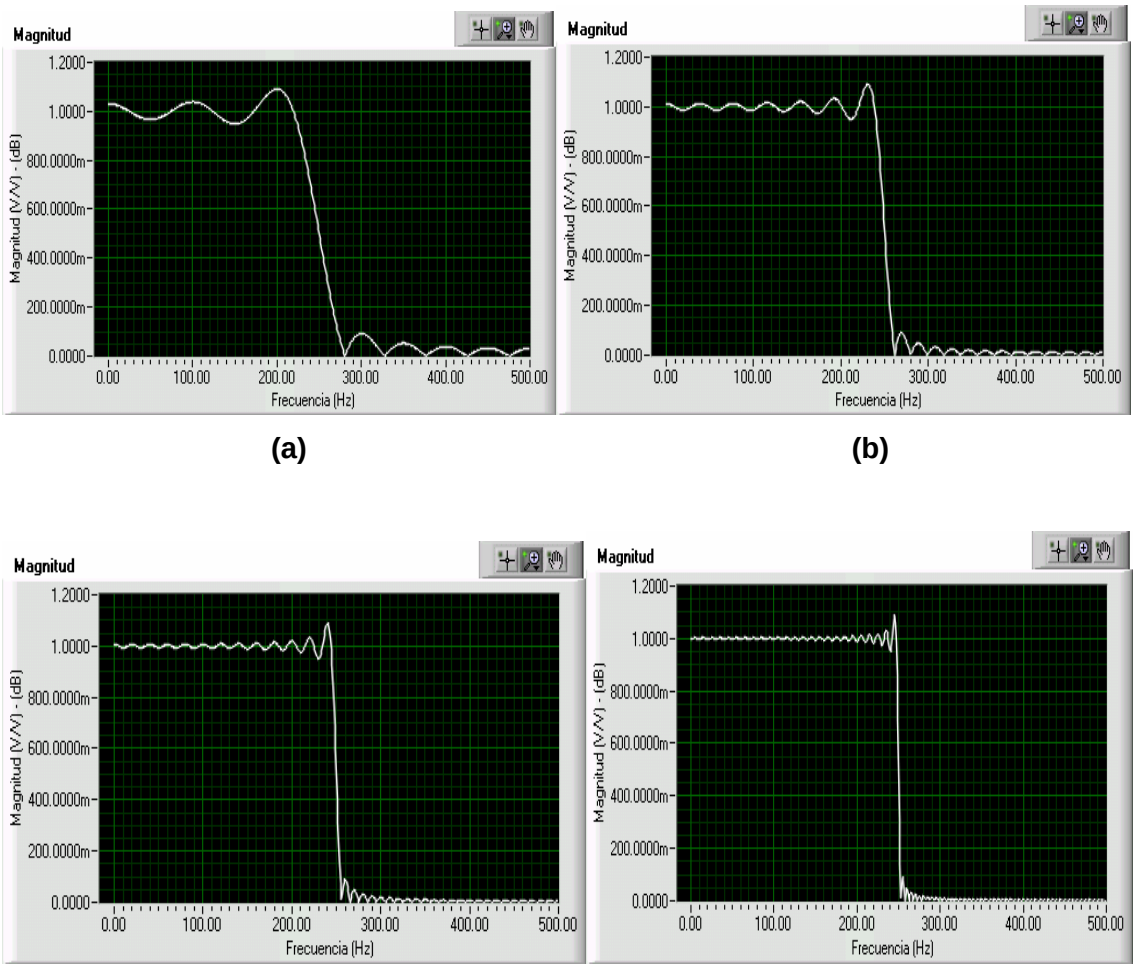


Figura 1.5. Efecto del truncamiento en un filtro pasa bajas con Fc = 250Hz y frecuencia de muestreo de 1kHz **a.** Orden 20, **b.** Orden 50, **c.** Orden 100, **d.** Orden 200

El diseño de un filtro realizable con función $H(e^{jw})$ puede definirse como el resultado de multiplicar los coeficientes, d_k, en cantidad infinita del filtro ideal por una ventana w[n] de (M+1) puntos con coeficientes w_k. Ello conlleva a que el filtro tendrá ahora una longitud finita de (M+1) coeficientes definidos por:

$$b_k = d_k\, w_k \qquad (1.23).$$

Entonces $H(e^{jw})$ que es la transformada de Fourier de h[n], es el resultado de la convolución periódica de la respuesta al impulso ideal deseada con la transformada de Fourier de la ventana $W(e^{jw})$ .Donde

$$W\left(e^{jw}\right)=\sum_{n=0}^{M}e^{-jwn}=\frac{1-e^{-jw(M+1)}}{1-e^{-jw(M+1)}}=e^{-jwM/2}\frac{sen\left[w\left(M+1\right)/2\right]}{sen\left(w/2\right)}. \quad (1.24).$$

La ventana rectangular es el típico caso de función de fase lineal generalizada con simetría. A medida que aumentamos M el ancho del lóbulo principal disminuye y su altura aumenta de forma que el área se mantiene, lo mismo sucede con los lóbulos laterales.

- **Ventanas Comúnmente Utilizadas**

  - ✓ Ventana Rectangular o Uniforme

    Este tipo de ventana permite reducir la zona de transición al aumentar el número de coeficientes, pero no tiene efecto sobre el rizado, sus coeficientes se definen según la relación:

    w[n] = 1, para 0 ≤ n ≤ M y cero en el resto (1.25).

  - ✓ Bartlett (triangular)

    2n/M, para 0 ≤ n ≤ M/2
    w[n] = 2-2n/M, para M/2 < n ≤ M (1.26).
    0, en el resto

  - ✓ Ventana Van Hann o Hanning

    La aplicación de esta ventana permite truncar gradualmente la respuesta ideal en lugar de un brusco corte de los coeficientes de Fourier del filtro. Es decir, si se van a desechar coeficientes, se hacen poco a poco y no bruscamente.

Los coeficientes de esta ventana están definidos por la relación:

$w[n] = 0.5 – 0.5 \cos(2\pi n/M)$, para $0 \leq n \leq M$ y cero en el resto (1.27).

- ✓ Ventana Hamming

Combina las características de la ventana Uniforme y la ventana Van Hann, entregando una mejor definición de los lóbulos laterales en la respuesta de frecuencia. Los coeficientes vienen dados por:

$w[n] = 0.54 – 0.46 \cos(2\pi n/M)$, para $0 \leq n \leq M$ y cero en el resto (1.28).

- ✓ Blackman

$w[n] = 0.42 – 0.5 \cos(2\pi n/M) + 0.08 \cos(4\pi n/M)$, para $0 \leq n \leq M$ y cero en el resto (1.29).

Al diseñar filtros FIR se desea obtener sistemas causales con respuesta de fase lineal generalizada, todas las ventanas están diseñadas con este propósito. Entonces

$w[n] = w[M-n]$, para $0 \leq n \leq M$ y cero en el resto. (1.30).

Es decir las ventanas son simétricas con respecto al punto M/2, entonces las transformadas de Fourier son de la forma

$$W(e^{jw}) = W_e(e^{jw})e^{-jwM/2} \quad (1.31).$$

Siendo $W_e(e^{jw})$ una función real y par de w,

El producto de la respuesta ideal deseada por la ventana da como resultado sistemas causales, ahora si la respuesta ideal deseada es también simétrica con respecto a M/2, entonces el sistema en general será causal y de fase generalizada, es decir

$$H_d(e^{jw}) = H_e(e^{jw})e^{-jwM/2} \qquad (1.32).$$

Y la respuesta del sistema

$$H(e^{jw}) = A_e(e^{jw})e^{-jwM/2} \qquad (1.33).$$

Siendo $A_e(e^{jw})$ una función real par de w e igual a la convolución periódica de las funciones reales de la respuesta ideal deseada $H_e(e^{jw})$ y de la ventana $W_e(e^{jw})$ respectivamente.

Tabla 1.3. Características generales de las ventanas

Ventana	Atenuación en lóbulo lateral [dB]	Máximo Atenuación de Rizado en la banda de rechazo [dB]	Ancho de Lóbulo Principal
Rectangular	-13	-21	$\frac{4\pi}{M+1}$
Hanning	-31	-44	$\frac{8\pi}{M}$
Hamming	-41	-53	$\frac{8\pi}{M}$
Blackman	-57	-74	$\frac{12\pi}{M}$

1.3.2 Diseño de Filtros FIR Mediante la Ventana de Kaiser

La relación entre la anchura del lóbulo principal y el área de los lóbulos laterales se ha estudiado ampliamente.

Kaiser descubrió que se puede formar una ventana cuasi-óptima utilizando la función de Bessel modificada de primera especie. La ventana de Kaiser se define como:

$$w[n] = \frac{I_0\left[\beta\left(1-\left[(n-\alpha)/\alpha\right]^2\right)^{1/2}\right]}{I_0(\beta)}$$, para 0 ≤ n ≤ M y cero en el resto (1.34).

Donde α = M/2 e $I_0(.)$ es la función de Bessel modificada de primera especie, definida por la serie:

$$I_0(X) = 1 + \sum_{k=1}^{\infty}\left(\frac{1}{k!}\left(\frac{X}{2}\right)^k\right)^2 ,$$ (1.35).

A diferencia de las otras ventanas, la ventana de Kaiser tiene dos parámetros: el orden del filtro, M, y el parámetro de forma β. Variando M y β se puede ajustar la amplitud de los lóbulos laterales y el ancho del lóbulo principal. Se ha demostrado que si se aumenta el orden del filtro M y β se mantiene constante, la banda de transición disminuye manteniéndose la amplitud de los lóbulos laterales. De hecho, Kaiser obtuvo, mediante amplias experimentaciones numéricas, una pareja de fórmulas que permiten al diseñador de filtros predecir los valores del orden del filtro y del factor de formas necesarias para cumplir unas determinadas especificaciones en frecuencia. Además, también demostró que, sobre un intervalo suficientemente amplio de condiciones, el nivel de rizado, δ está determinado por la selección de β. Suponiendo fijo el nivel de rizado, la frecuencia de corte de la banda de paso de un filtro paso bajo wp se define como la máxima frecuencia para la que $|H(e^{jw})| \geq 1-\delta$. La frecuencia de corte de la banda eliminada w_S, se define como la mínima frecuencia para la que $|H(e^{jw})| \leq \delta$. Por otro lado, la anchura normalizada de la región de transición es:

$$\Delta w = \frac{w_P - w_S}{2w_n}$$ (1.36).

Donde w_n es la frecuencia angular de Nyquist. Para la aproximación del filtro paso bajo se define.

$$A = -20 \log_{10} \delta \qquad (1.37).$$

Kaiser determinó empíricamente que el valor de β necesario para cumplir un valor específico de A está dado por:

$$\beta = \begin{cases} 0.1102(A-8.7), & \text{para } A>50 \\ 0.5842(A-21)^{0.4}+0.07886(A-21), & \text{para } 21 \le A \le 50 \\ 0, & \text{para } A < 21 \end{cases} \qquad (1.38).$$

Para el caso de que β sea cero, la ventana es igual a la rectangular. Además, Kaiser demostró que para cumplir unos valores específicos de A, Δw y M se debe satisfacer:

$$M = \frac{A-8}{2.285\Delta w} \qquad (1.39).$$

La Ecuación (1.39) permite predecir el valor de M con una precisión de ±2 para un amplio margen de valores de A y Δw. Por tanto, con estas fórmulas, el método de diseño basado en la ventana de Kaiser casi no requiere iteraciones de prueba y error. Las características generales en una ventana Kaiser para distintos parámetros se pueden ver en la Tabla 1.4.

Con el uso de las fórmulas de diseño de la ventana de Kaiser, es inmediato diseñar un filtro FIR paso bajo que cumpla unas determinadas especificaciones. El procedimiento de diseño general con ventanas sigue los siguientes pasos:

- Convertir las especificaciones del filtro deseado para prototipo de filtro pasa bajos y se elige el tipo de ventana que puede cumplir con dichas condiciones, teniendo en cuenta los valores típicos de cada ventana que se observan en la Tabla 1.3 Por

ejemplo una ventana Rectangular no puede tener un rizado con una atenuación superior a 21dB en la banda de rechazo según lo indica la Tabla 1.3

- Determinar la frecuencia de corte del filtro paso bajo ideal. Que debido a la simetría de la aproximación en la discontinuidad de $H(e^{jw})$, es:

$$w_c = \frac{w_p + w_s}{2} \qquad (1.40).$$

- Estimar el orden del filtro usando la Ecuación (1.39) para el cálculo del orden según Kaiser, seguidamente determinar mediante la experimentación si las especificaciones se cumplen, de lo contrario se aumenta el orden del filtro hasta cumplir las condiciones requeridas.

- Determinar la secuencia de ponderación del filtro ideal pasa bajas por medio de:

$$h_d[n] = \frac{sen[w_c(n - M/2)]}{\pi(n - M/2))} \quad (1.41).$$

- Calcular los parámetros de la ventana elegida, su correspondiente secuencia y determinar la correspondiente respuesta al impulso del filtro como el producto de la secuencia del filtro ideal por la secuencia de la función de la ventana

$$h[n] = \frac{sen[w_c(n - M/2)]}{\pi(n - M/2))} w[n] \quad (1.42).$$

- Por ultimo, si el filtro selectivo no es paso bajo se realiza la transformación al tipo de filtro deseado según la Tabla 1.5.

Tabla 1.4. Características generales en una ventana Kaiser para distintos parámetros

Parámetro β	Atenuación en lóbulo lateral [dB]	Ancho de Banda del lóbulo principal [dB]	Máxima Atenuación de Rizado en la banda de rechazo [dB]
2.0	-19	$\frac{3\pi}{M+1}$	-29
3.0	-24	$\frac{4\pi}{M+1}$	-37
4.0	-30	$\frac{5.2\pi}{M+1}$	-45
5.0	-37	$\frac{6.4\pi}{M+1}$	-54
6.0	-44	$\frac{7.6\pi}{M+1}$	-63
7.0	-51	$\frac{9\pi}{M+1}$	-72
8.0	-59	$\frac{10.2\pi}{M+1}$	-81
9.0	-67	$\frac{11.4\pi}{M+1}$	-90
10.0	-74	$\frac{12.8\pi}{M+1}$	-99

- **Transformaciones de los Filtros**

Aunque el método expuesto se ha centrado en los filtros paso bajo, también es posible diseñar los otros tipos de filtros selectivos, paso alto, pasa banda y rechaza banda. Para ello se podría utilizar el mismo procedimiento que se vio anteriormente, pero en vez de aplicar la banda pasante a las bajas frecuencias se puede realizar para cualquier otra forma de banda pasante. Sin embargo, y de igual manera que con los filtros analógicos, aquí también existen transformaciones de paso bajo a cualquiera de los otros tipos de filtros.

Así por ejemplo, si $h_{LP}[n]$ son los coeficientes de la respuesta al impulso de un filtro paso bajo con una frecuencia de corte wc, entonces los coeficientes de un filtro pasa altas con una frecuencia de corte de $\pi - w_C$ se pueden calcular con la Ecuación 1.43.

$$h_{HP}[n]= (-1)^n h_{LP}[n] \qquad (1.43)$$

En la Tabla 1.5 se muestran otras transformaciones de filtros útiles.

Tabla 1.5. Conversión entre tipos de filtros diseñados por ventanas

Conversión	Transformación	Parámetros
A paso alto	$h_{HP}[n] = (-1)^n h_{LP}[n]$	$(w_C)_{HP} = \pi - (w_C)_{LP}$
A paso banda	$h_{BP}[n] = (2\cos(nw_0))h_{LP}[n]$	w_0 = Es la frecuencia central $w_1 = w_0 - (w_C)_{LP}$ $w_2 = w_0 + (w_C)_{LP}$
A rechaza banda	$h_{Bs}[n] = \delta[n] - h_{BP}[n]$	Las mismas de pasa banda

Ejemplo 1.1

Diseñar un filtro FIR pasa bajas de orden 2 utilizando ventana rectangular, con frecuencia de corte 100Hz y frecuencia de muestreo 1kHz y graficar su respuesta en frecuencia. Para observar resultados, procesar una señal senoidal de 200Hz utilizando el filtro diseñado.

Solución:

1. Datos suministrados:

f_c=100 (Hz). Ω_c=2*π*100=200*π (rad/s).

f_s=1000 (Hz).

$x_{ent}(t)$=sen(400*π*t)

2. Se calcula la frecuencia de corte digital como w=ΩT, donde T=1/ f_s. Entonces w_c=200*π*(1/1000)=0.2*π (rad/muestra)

3. De acuerdo a la Ecuación 1.41 los coeficientes del filtro pasa bajas ideal para los datos suministrados son $h_d[n] = \frac{sen[0.2\pi(n-1)]}{\pi(n-1))}$ como se ilustran en la Figura 1.6.

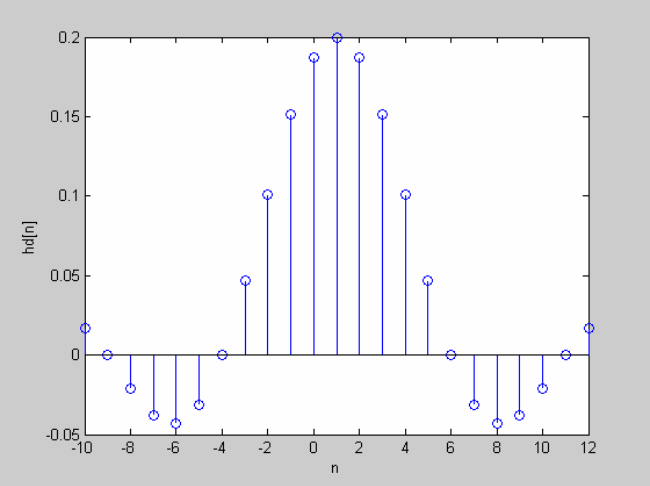


Figura 1.6 Respuesta al impulso del filtro ideal pasa bajas

De la Figura 1.6 se observa la simetría con respecto al punto M/2=1 de la respuesta al impulso del filtro ideal con fase lineal generalizada tipo I.

4. La ventana rectangular que se debe utilizar es: w[n]=1, para 0≤n≤2 y cero en el resto. Esta ventana se ilustra en la Figura 1.7.

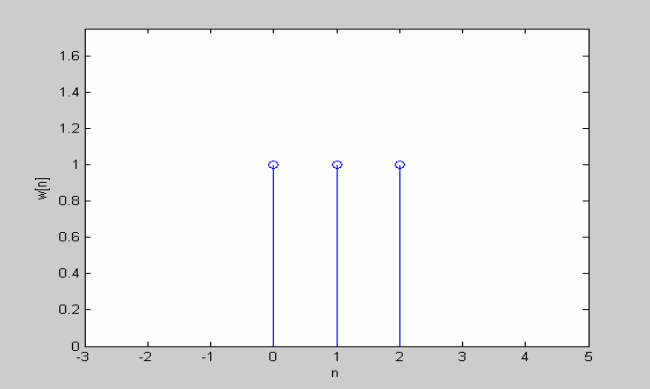


Figura 1.7 Ventana rectangular de orden dos

5. De acuerdo a la Ecuación 1.42 los coeficientes del filtro FIR causal realizable son:

$$h[n] = \frac{sen[0.2\pi(n-1)]}{\pi(n-1)}, \quad 0 \leq n \leq 2$$

$$h[n] = 0, \text{ en el resto.}$$

De acuerdo a esta función:

h[0]=0.18709

h[1]=0.2

h[2]=0.18709

Estos coeficientes se observan en la Figura 1.8

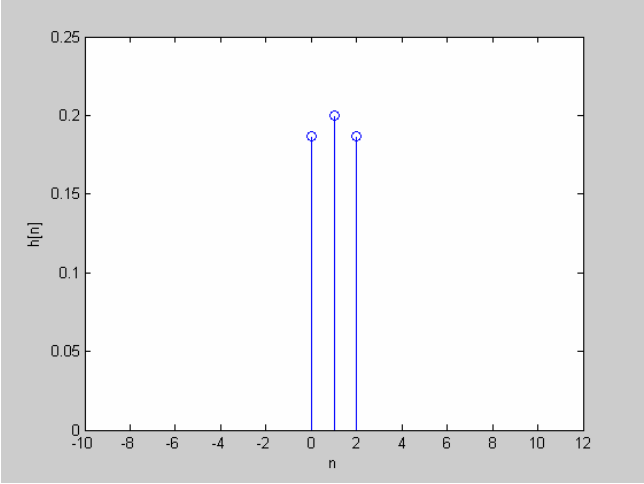


Figura 1.8 Respuesta al impulso del filtro FIR pasa bajas causal.

Por ser un filtro FIR la respuesta al impulso proporciona directamente los coeficientes de la ecuación en diferencia que representa al sistema del filtro pasa bajas. La ecuación sería:

$$y[n]=0.18709x[n]+0.2x[n-1]+0.18709x[n-2] \qquad (1.44)$$

Según la Ecuación 1.3 la función de transferencia del filtro es:

$$H(z)=0.18709+0.2z^{-1}+0.18709z^{-2} \quad (1.45)$$

Solución en MATLAB

>> freqz([0.18709 0.2 0.18709],1,100)

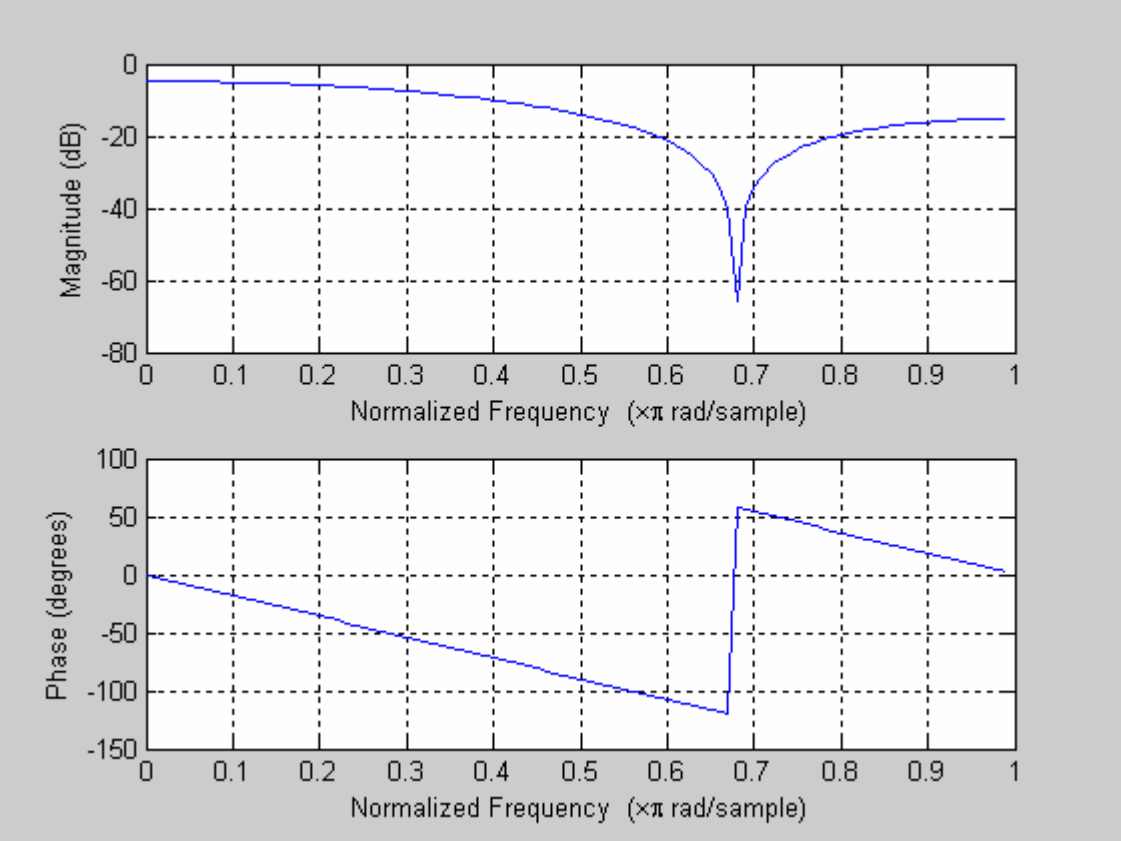


Figura 1.9 Magnitud Lineal y Fase de la respuesta en frecuencia del filtro FIR del Ejemplo 1.1.

A partir de la ecuación en diferencias de la Ecuación 1.44 se puede calcular la salida del filtro para cualquier secuencia de entrada de forma no recursiva. Este proceso se puede hacer manualmente, pero es un proceso tedioso si la secuencia de entrada es larga. Para procesar una secuencia de un gran número de muestras se puede programar un algoritmo en LabVIEW que realice este procedimiento. En la Figura 1.10 se observa el algoritmo que representa la ecuación de diferencias de la Ecuación 1.44. Se observa de la figura que el algoritmo de cómputo es básico, ya que se implementa con multiplicadores,

sumadores y registros de desplazamientos que realizan la función de retardo. El ciclo *WHILE* principal se ejecuta hasta procesar el número de muestras total que componen la secuencia de entrada.

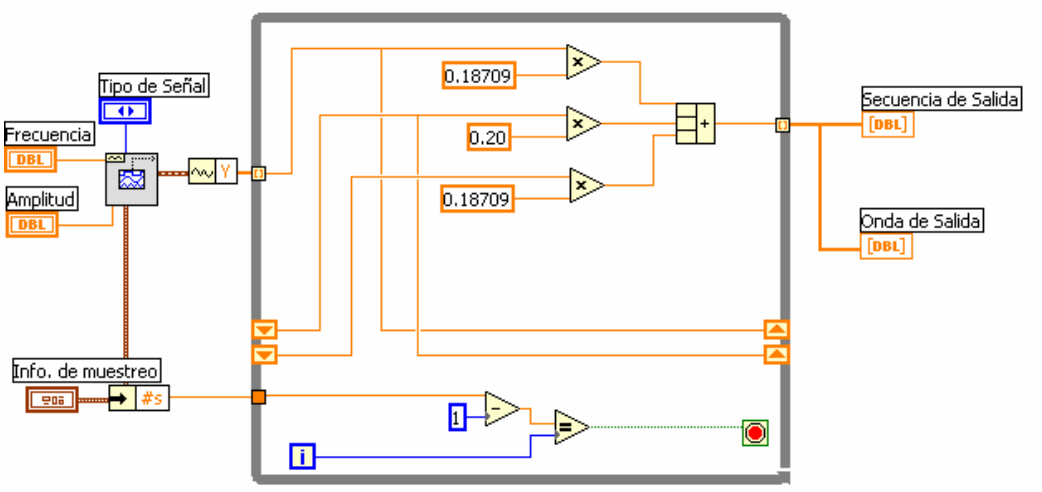


Figura 1.10 Programación en LabVIEW de un filtro FIR pasa bajas de orden 2 con frecuencia de corte 100Hz y ventana rectangular

En la Figura 1.11 se observa el procesamiento de la señal senoidal de 200Hz por medio del algoritmo programado en LabVIEW en la Figura 1.10. Nótese que la amplitud de la secuencia de salida está atenuada en 0.3V, lo cual se podía predecir observando la magnitud de la respuesta en frecuencia del filtro pasa bajas observada en la Figura 1.9.

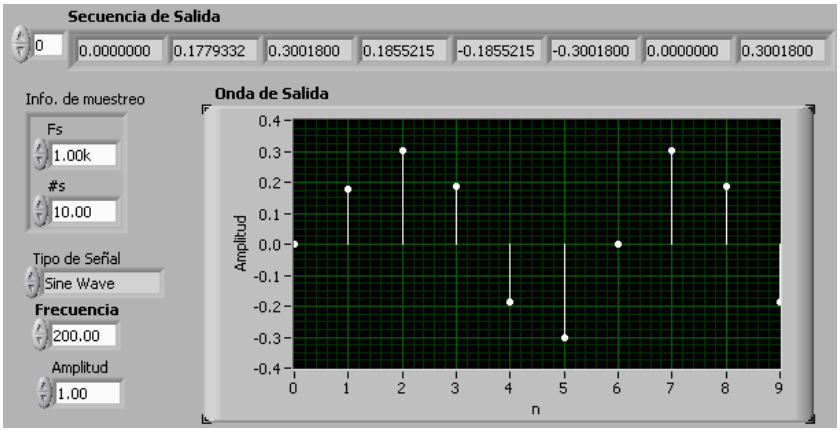


Figura 1.11 Secuencia de Salida del filtro para una entrada senoidal de 200Hz, muestreada a 1kHz.

En general el proceso para diseñar un filtro digital y procesar una secuencia de entrada con el filtro diseñado se puede apoyar en software de aplicación como LabVIEW o MATLAB para agilizar el procedimiento. En el caso específico del software de LabVIEW, se cuenta con una variedad de instrumentos virtuales para el procesamiento digital que son las herramientas poderosas que se utilizan en el capítulo 3 para el diseño e implementación de filtros digitales.

Ejemplo 1.2

Diseñar un filtro pasa bajas con ventana de Kaiser con frecuencia de paso de 6kHz y frecuencia de rechazo de 9kHz, sabiendo que se admite un rizado en la banda de paso de 0.1 y en la supresora es de 0.01. La frecuencia de muestreo de 30kHz.

Como el método de diseño de la ventana implica que los rizados deben ser idénticos tanto en la banda de paso como en la banda de rechazo entonces se elige el rizado más pequeño, esto es, se toma $\delta_2 = 0.01$.

1. El primer paso consiste en convertir las especificaciones del filtro a un prototipo de filtro pasa bajas y elegir la ventana que pueda cumplir con los requerimientos. Sin embargo, en este ejemplo se obvia este paso ya que el enunciado del ejemplo exige diseñar un filtro pasa bajas con ventana Kaiser.

2. Se determina la frecuencia de corte del filtro paso bajo ideal, que debido a la simetría en la discontinuidad de la respuesta en frecuencia del ideal, sería:

$$\Omega_c = 2\pi\left(\frac{6000+9000}{2}\right) = 2\pi 7500 \text{ rad/s (1.46).}$$

3. Se calculan el ancho de la banda de transición normalizado y la atenuación del rizado como indican las Ecuaciones (1.36). Y (1.37)

$$\Delta\omega = 2\pi\left(\frac{9000-6000}{30000}\right) = 0.2\pi \quad (1.47).$$

$$A = -20 \log_{10}(\delta_2) = 40dB \quad (1.48).$$

4. Los resultados de las Ecuaciones (1.46) y (1.47) se sustituyen en las Ecuaciones (1.38) y (1.39) para obtener el orden y el factor de forma respectivamente.

$$M = \frac{40-8}{2.285\Delta\omega} = 22.2886 \qquad (1.49).$$

$$\beta = 0.5842(40-21)^{0.4} + 0.07886(40-21) = 3.3953 \qquad (1.50).$$

Redondeando los valores de M y β se tiene:

$$M = 22 \text{ y } \beta = 3.4 \quad (1.51).$$

5. La respuesta al impulso del filtro pasa bajo ideal se calcula como lo indica la Ecuación (1.52).

$$h_d[n] = 0.5\frac{sen[(n-11)0.5\pi]}{(n-11)0.5\pi} \quad (1.52).$$

La respuesta al impulso del filtro causal realizable viene dada por la Ecuación (1.53)

$$h[n] = 0.5\frac{sen[(n-11)0.5\pi]}{(n-11)0.5\pi} * w[n] \quad (1.53)$$

Donde w[n] son los coeficientes de la ventana de Kaiser dados por la Ecuación (1.34).

SOLUCIÓN EN MATLAB

» fp=6e3;fr=9e3;fs=30e3;d=0.01;%se define frecuencia de paso, rechazo, muestreo y rizado

» A=-20*log10(d);Aw=2*pi*(fr-fp)/fs;w_cn=(fp+fr)/(fs);% se calculan parámetros de Kaiser

» M=round((A-8)/(2.285*Aw));%se calcula orden del filtro

» beta=.5842*((A-21)^.4)+0.07886*(A-21);%se calcula parámetro de forma

» b=fir1(m,w_cn,kaiser(m+1,beta));%se calcula coeficientes del filtro

» [H,F]=freqz(b,1,2048,fs);

» plot(F,abs(H))

» plot(F,angle(H))

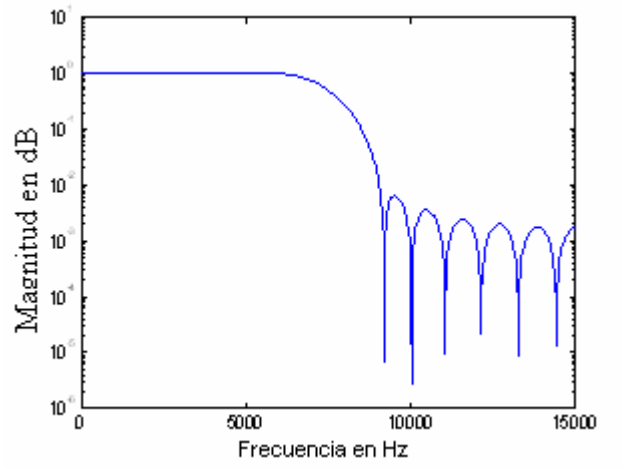


Figura 1.12 Magnitud del filtro del ejemplo 1.2.

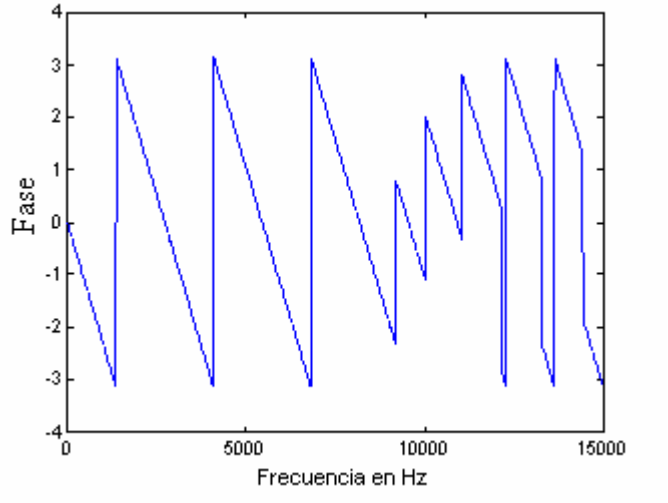


Figura 1.13 Fase del Ejemplo 1.2.

1.3.3 Diseño de Filtros FIR Óptimos de Amplitud de Rizado Constante

Se desea tener un método de diseño de filtros FIR que permita tener un mayor control sobre los errores permitidos en las bandas de paso y rechazo respectivamente, y que este error sea mínimo. Este es el punto de partida del estudio de los filtros FIR óptimos. Para el caso de filtros FIR usando ventanas ya se vio que producen generalmente los mismos errores en la banda de paso y en la banda eliminada, además también debido a la simetría de las ventanas debemos diseñar los filtros con sobradas especificaciones tomando el menor rizado de las bandas.

Entonces el diseño de filtros óptimos busca tener control sobre todos los parámetros de diseño como son orden del filtro, rizados en las bandas pasantes y de rechazo, frecuencias de corte y frecuencias de paso, por tal motivo se han desarrollado una serie de algoritmos de diseño en los que se fijan algunos parámetros y, mediante procedimientos iterativos, se obtienen ajustes para los restantes parámetros, por ejemplo. Parks y McClellan, han desarrollado un método que deja fijo el orden del filtro y las frecuencias de paso y rechazo respectivamente, mientras se varía la relación δ_1/δ_2 , y δ_1 o δ_2 , que es el método dominante en el diseño óptimo de filtros FIR.

El algoritmo de Parks – McClellan se basa en plantear el problema de diseño de filtros en un problema de aproximación de polinomios. Considérese un filtro FIR de simetría par como el de la Ecuación (1.15).

Entonces la respuesta en frecuencia es.

$$A_e\left(e^{jw}\right)=\sum_{n=0}^{M}h_e[n]e^{-jwn} \quad (1.54).$$

Donde M es el orden del filtro. Desarrollando la Ecuación (1.54).se tiene

$$A_e\left(e^{jw}\right)=h_e[0]+\sum_{n=1}^{L}2h_e[n]\cos(wn) \quad (1.55).$$

Donde L es M/2 y $A_e(e^{jw})$ es una función real, par y periódica de w, como el sistema que representa la Ecuación (1.54) es de fase generalizada entonces podemos decir que la respuesta en frecuencia del filtro es

$$H(e^{jw}) = A_e(e^{jw})e^{-jwM/2} \quad (1.56).$$

En particular el teorema de Parks – McClellan dice que los términos cos(wn) se pueden expresar como sumas de potencias del cos(w) así:

$$\cos(wn) = T_n(\cos w) \quad (1.57).$$

Donde $T_n(x)$ es un polinomio de grado n, entonces remplazando la Ecuación (1.57). en la Ecuación (1.55). Se tiene

$$A_e\left(e^{jw}\right) = \sum_{K=0}^{L} a_K \left(\cos w\right)^K \quad (1.58)$$

También se puede expresar así

$$A_e(e^{jw}) = P(x)|_{x=\cos w} \quad (1.59).$$

$$\text{Donde } P(x) = \sum_{K=0}^{L} a_K x^K \quad (1.60).$$

El algoritmo de Parks – McClellan como ya se menciono, plantea el problema a una aproximación de una función que minimiza el error, para esto definimos la función error así:

$$E(w) = W(w)[H_d(e^{jw}) - A_e(e^{jw})] \quad (1.61).$$

Donde para el caso de un filtro pasa bajas se define

$$H_d(e^{jw}) = \begin{cases} 1, & 0 \le w \le w_p \\ 0, & w_s \le w \le \pi \end{cases} \quad (1.62).$$

$$W(w) = \begin{cases} \frac{1}{K}, & 0 \le w \le w_p \\ 1, & w_s \le w \le \pi \end{cases} \quad (1.63).$$

Parks y McClellan definieron que para aproximar a la minimización de la Ecuación (1.61) es necesario que la función P(x), como se define en las ecuaciones (1.59) y (1.60), cumpla con el teorema de alternación que dice:

- Para el único polinomio de grado r que minimiza la función de error definida por la Ecuación (1.61) se cumple que debe tener al menos (r+2) alternancias de los valores máximos de error permitidos.
- Las alternancias deben cambiar consecutivamente de signo.

La función de error se puede re definir de la siguiente manera

$$E_p(x) = W_p(x)[D_p(x) - P(x)] \quad (1.64)$$

Donde $D_p(x)$ es una función deseada de x y $W_p(x)$ es una función de peso

Para el caso de los filtros simétricos de orden par, haciendo x= cosw y r=L, se tiene

$$P(\cos w) = \sum_{K=0}^{L} a_K (\cos w)^K \quad (1.65)$$

$D_p(x)$ la respuesta deseada para el caso de un filtro pasa bajas como el de la Ecuación (1.62) es

$$D_p(\cos w) = \begin{cases} 1, & \cos w_p \le \cos w \le 1, \\ 0, & -1 \le \cos w \le \cos w_s \end{cases} \qquad (1.66).$$

$$W_p(\cos w) = \begin{cases} \frac{1}{K}, & \cos w_p \le \cos w \le 1, \\ 1, & -1 \le \cos w \le \cos w_s \end{cases} \qquad (1.67)$$

Y el error queda

$$E_p(\cos w) = W_p(\cos w)[D_p(\cos w) - P(\cos w)] \qquad (1.68).$$

En resumen el diseño de un filtro óptimo de rizado constante varía los parámetros de la función de error de la Ecuación (1.68), dependiendo del tipo de filtro que se desee diseñar se realizas el siguiente procedimiento para un filtro pasa bajas.

1. Se define el orden del filtro M, mediante la siguiente expresión

$$M = \frac{-10\log(\delta_1\delta_2) - 13}{2.324\Delta w} \quad (1.69)$$

2. Sespués se calculan los parámetros b_K, W_K, d_K, C_K por medio de las Ecuaciones (1.70) donde $x_i = \cos wi$

$$b_K = \prod_{i=1}^{L+2} \frac{1}{(x_K - x_i)} \quad (1.70a).$$

$$W(w_K) = \frac{1}{K}$$, para 0 ≤ w ≤ w_p y cero en el resto (1.70b).

$$C_K = H_d\left(e^{jwK}\right) - \frac{(-1)^{K+1}\delta}{W(w_K)} \quad (1.70c)$$

$$d_K = \prod_{i=1}^{L+1} \frac{1}{(x_K - x_i)} = b_K\left(x_K - x_{L+2}\right) \quad (1.70d).$$

3. Por último se calcula la respuesta óptima del filtro por medio de

$$A_e(e^{jw}) = P(\cos w) = \frac{\sum_{K=1}^{L+1}\left[d_K/(x - x_K)\right]C_K}{\sum_{K=1}^{L+1}\left[d_K/(x - x_K)\right]} \quad (1.71).$$

Debido a la complejidad computacional de las fórmulas utilizadas en el algoritmo iterativo de Parks-McClellan, se explicará con un ejemplo los pasos necesarios para diseñar un filtro utilizando MATLAB. El diseño de este filtro en MATLAB es un proceso de dos pasos. Primero se debe estimar el orden del filtro resultante del filtro FIR óptimo Parks-McClellan que cumple las especificaciones de diseño utilizando el comando *remezord*. La sintaxis del comando es: [n,fo.wo,w]=remezord(f,m,dev), donde f es un vector con las frecuencias que limitan las bandas del filtros (se pueden diseñar filtros multibanda), por tanto para un filtro pasa bajas tendría la forma [w_P w_S]. El vector m contiene la magnitud de la respuesta en frecuencia ideal en cada banda, por tanto para un filtro pasa bajas sería [1 0]. El vector dev contiene la amplitud del rizado en cada banda, por tanto para un filtro pasa bajas sería [δ_1 δ_2]. El valor de w es la frecuencia de muestreo. Luego de tener fo, mo y n se ingresan al comando *remez* de la siguiente manera: b=remez(n,fo,mo) donde el vector b contiene los coeficientes del filtro FIR óptimo diseñado con la siguiente transformada Z:

$$H(z) = b(1) + b(2)z^{-1} + b(3)z^{-2} + \cdots + b(n+1)z^{-n}$$

Ejemplo 1.3 Diseño de un filtro pasa bajas utilizando el algoritmo de Parks-McClellan

Se desea diseñar un filtro pasa bajas con una frecuencia de pasa banda de 1500Hz, una frecuencia de rechazo de 2000Hz, un rizado en pasa banda de 0.01, un rizado en la banda de rechazo de 0.1, y una frecuencia de muestreo de 8000Hz.

Solución en Matlab

```
» [n,fo,mo,w] = remezord( [1500 2000], [1 0], [0.01 0.1], 8000 );
» b = remez(n,fo,mo,w);
» freqz(b,1);
```

En la Figura 1.14 se observan la magnitud en (dB) de la respuesta en frecuencia del filtro y la fase continua. Se observa que el rizado es constante en la banda rechazo y la fase es lineal.

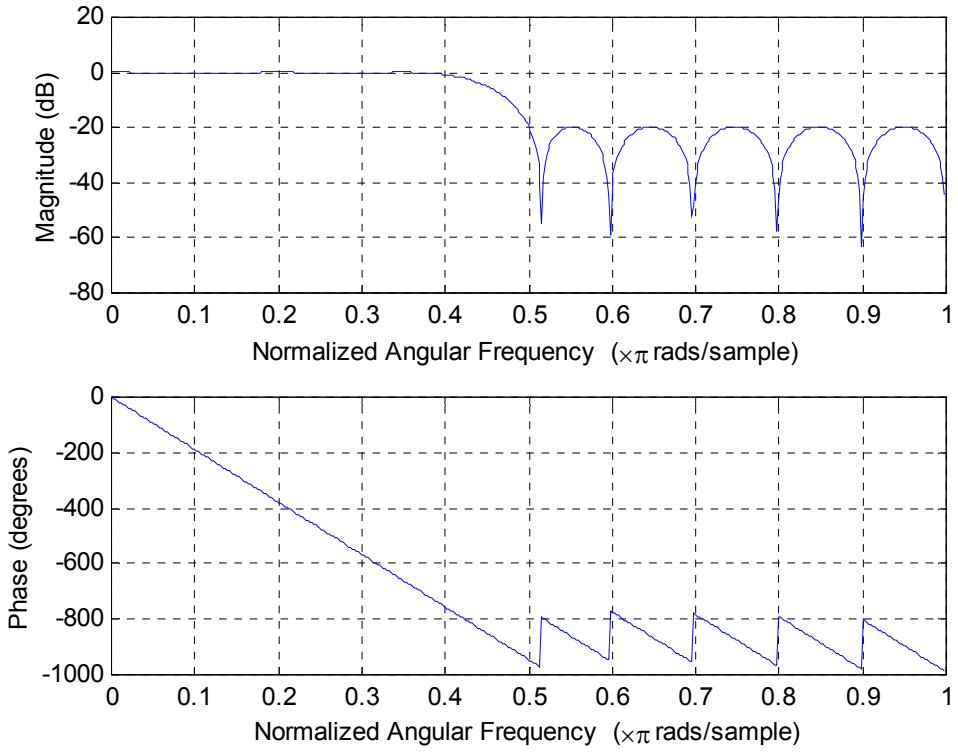


Figura 1.14 Magnitud y Fase del filtro óptimo FIR diseñado con el algoritmo de Parks-McClellan.

1.4 DISEÑO DE FILTROS RECURSIVOS IIR

El método tradicional de diseño de filtros IIR en tiempo discreto se basa en la transformación de un filtro analógico en un filtro digital que cumpla las especificaciones preestablecidas. Esta solución es razonable por varios motivos:

1. El diseño de filtros IIR analógicos está muy avanzado, y como se pueden obtener resultados útiles, es ventajoso utilizar los procedimientos de diseño que ya se han desarrollado para los filtros en tiempo continuo.

2. Muchos métodos útiles de diseño de filtros IIR en tiempo continuo dan como resultado fórmulas de diseño simples en forma cerrada. Los métodos de filtros IIR digitales que se basan en esas fórmulas estándar de diseño de filtros IIR continuos son fáciles de realizar.

El hecho de que los diseños de filtros en tiempo continuo se puedan trasladar a diseños de filtros digitales no quiere decir que tengan la misma respuesta en frecuencia. Generalmente sucede que el filtro analógico empleado para la aproximación tiene una respuesta en frecuencia diferente de la respuesta en frecuencia efectiva del filtro digital. Esta circunstancia indica que al diseñar un filtro digital se parte de un conjunto de especificaciones en tiempo discreto; mientras las características del filtro en tiempo continuo se obtienen de la transformación. Al realizar esta conversión se desea que la respuesta en frecuencia del filtro digital conserve las propiedades esenciales del filtro analógico. Esto implica concretamente que se espera que el eje imaginario del plano s se transforme en la circunferencia unidad del plano z. Una segunda condición es que un filtro estable analógico se debe transformar en un filtro estable de tiempo discreto. Esto significa que si el filtro continuo tiene los polos en el semiplano negativos de s, el filtro digital tiene que tener los polos dentro del círculo unidad del plano z, tal como se ilustra en la Figura 1.16.

.

Estas restricciones son básicas para las técnicas de diseño de los filtros digitales IIR.

1.4.1 Diseño de Filtros IIR Mediante Transformadas Bilineales

Los filtros recursivos pueden ser diseñados por varios métodos, siendo el más común el basado en las transformaciones bilineales. Este procedimiento requiere del conocimiento de la función de transferencia del filtro que se desea diseñar en el tiempo continuo. Los coeficientes del filtro en el dominio (s) son transformados a uno equivalente en el dominio z.

Los coeficientes que surgen de la transformación forman los coeficientes del filtro IIR. El origen de este método viene dado por la cantidad de experiencia acumulada en el diseño de filtros analógicos. Por tanto, todos los polinomios, tablas, métodos analíticos y gráficos para definir el filtro analógico, serán usados en el diseño de los filtros recursivos.

Esta transformación se define como:

$$S = \frac{2}{T}\frac{1 - Z^{-1}}{1 + Z^{-1}} = \frac{2}{T}\frac{Z - 1}{Z + 1} \quad (1.72).$$

Y su relación inversa es:

$$Z = \frac{1 + \left(\frac{T}{2}\right)S}{1 - \left(\frac{T}{2}\right)S} \quad (1.73).$$

Ahora sustituimos S = σ + jΩ en la Ecuación (1.73) tenemos

$$Z = \frac{1 + \sigma\left(\frac{T}{2}\right) + j\Omega\left(\frac{T}{2}\right)}{1 - \sigma\left(\frac{T}{2}\right) - j\Omega\left(\frac{T}{2}\right)} \quad (1.74)$$

De la Ecuación (1.74) se observa que si σ < 0 entonces |Z| < 1 y si σ > 1 entonces |Z| > 1, para todo valor de Ω. Es decir, si los polos del filtro analógico están en el semiplano izquierdo de s, su imagen en el plano *z* está en el interior de la circunferencia unidad. Por tanto, los filtros en tiempo continuo causales y estables se transforman en filtros en tiempo discreto causales y estables. Seguidamente, para demostrar que el eje jΩ se transforma en la circunferencia unidad se procede a sustituir s = jΩ en la Ecuación (1.73), con lo que se obtiene:

$$Z = \frac{1 + j\Omega\frac{T}{2}}{1 - j\Omega\frac{T}{2}} \quad (1.75).$$

En la Ecuación (1.75) se observa que |z| es uno para cualquier valor de Ω.

De hecho, para obtener la relación de sus respuestas en frecuencias se sustituye s por jΩ y z por e^{jw} en la Ecuaciones (1.72) y (1.73) dando como resultado:

$$\Omega = \frac{2}{T}\tan\left(\frac{w}{2}\right) \quad (1.76).$$

$$w = 2\arctan\left(\frac{\Omega T}{2}\right) \quad (1.77).$$

Hay que notar cómo el intervalo de la frecuencia digital 0 ≤ w ≤ $\frac{\pi}{T}$ se transforma en el intervalo de frecuencia analógica 0 ≤ Ω ≤ ∞ como se ilustra en la Figura 1.15.

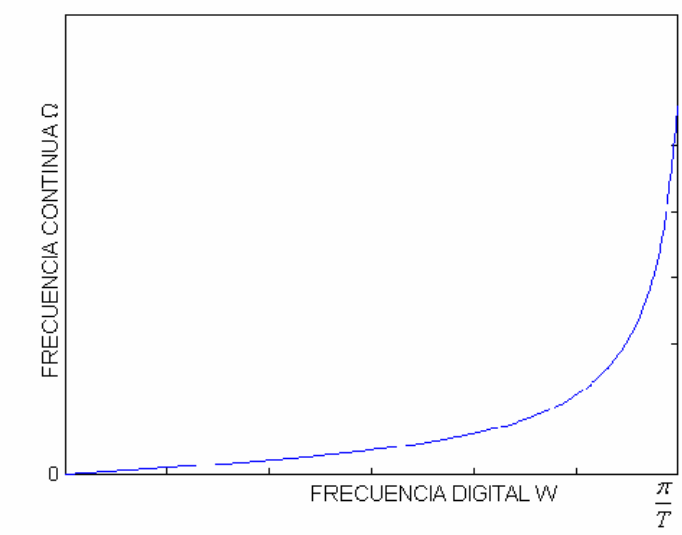


Figura 1.15 Relación entre la frecuencia digital-análoga

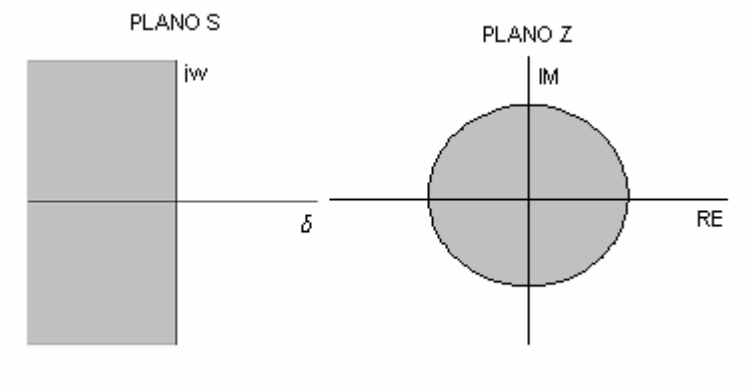


Figura 1.16 Transformación bilineal del plano (s) al plano (z)

El diseño de filtros IIR aplicando transformaciones bilineales empieza normalmente con la especificación de los requisitos del filtro en el dominio de la frecuencia digital (w). Estas especificaciones de frecuencias digitales son transformadas a frecuencias continuas mediante la Ecuación (1.76), que permiten obtener las especificaciones del filtro analógico

El paso siguiente consiste en elegir la topología del filtro analógico que más convenga, esto depende de la aplicación en particular. Teniendo el diseño del filtro analógico solo resta por hacer la transformación de (s) a (z) mediante la Ecuación (1.72).

En resumen el diseño de un filtro IIR sigue los siguientes pasos

1. Definir características del filtro digital.
2. Realizar la operación de acuerdo con la Ecuación (1.76) obteniendo las frecuencias analógicas Ω
3. Diseñar el filtro analógico con las frecuencias definidas en el punto 2.
4. Reemplazar (s) en el filtro analógico por la expresión dada en la Ecuación (1.72).

Ejemplo 1.4 Diseño de un filtro IIR

Diseñar un filtro digital paso bajo IIR con una frecuencia de muestreo de 10kHz, con un a frecuencia de corte de $\frac{\pi}{5}$ y, al menos, una atenuación de 10 dB a la frecuencia de $\frac{2\pi}{5}$

Solución: Siguiendo la metodología se transforman las frecuencias digitales a frecuencias analógicas con la Ecuación (1.76)

$$f_C = 1034Hz \text{ , } f_S = 2312Hz$$

Para este ejemplo se utilizará la topología Butterworth cuyas ecuaciones de diseño se encuentran en muchos textos de filtrado analógico. Sin embargo, el diseño se podría realizar utilizando ecuaciones de diseño de topologías Chebyshev, elíptica o Bessel.

Se parte de una frecuencia de corte 1034Hz y frecuencia de rechazo 2312Hz con atenuación de 10dB.

Utilizando las ecuaciones de diseño para el orden de un filtro de Butterworth, resulta n = 1.368 por lo que se debe tomar el valor entero superior, esto es, orden del filtro 2. La función de transferencia en el dominio (s) es:

$$H(s) = \frac{1}{\left(\frac{s}{2\pi 1034}\right)^2 + 1.41\left(\frac{s}{2\pi 1034}\right) + 1}$$

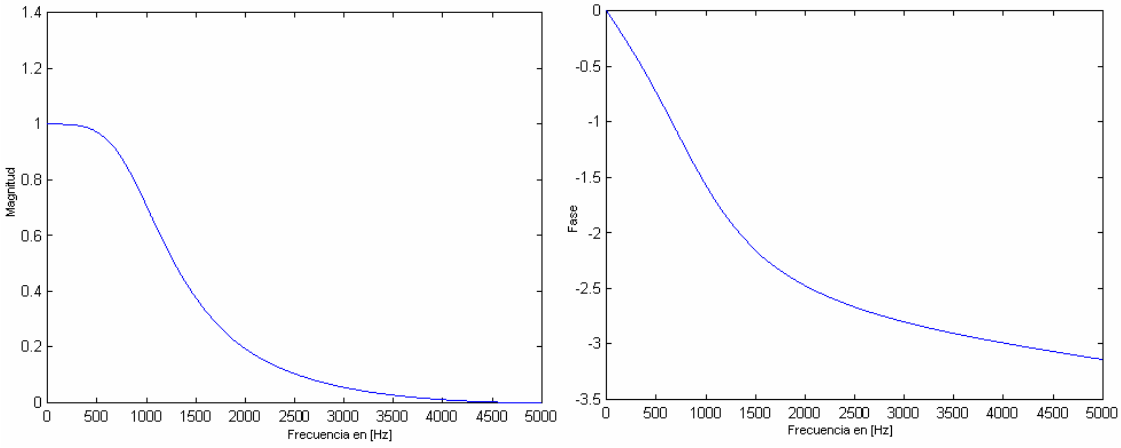
Aplicando la Ecuación (1.72) se transforma H(s) en H(z)

$$H(Z) = \frac{1 + 2Z^{-1} + Z^{-2}}{14.81 - 16.95Z^{-1} + 6.13Z^{-2}}$$

La respuesta en frecuencia de H(z) es como se ilustra en las Figuras (1.17)

Solución en Matlab

```
» wd=[pi/5 2*pi/5];%frecuencias digitales
» fa=1/(pi*T)*tan(wd/2);%calculo de frecuencias analógicas
» T=1e-4;fn=1/(2*T);%periodo de muestreo y frecuencia normalizada
» n=buttord(2*pi*fa(1),2*pi*fa(2),3,10,'s');%orden del filtro
» [p,q]=butter(n, 2*pi*fa(1),'s');%calculo de polos y ceros de la función de transferencia
» g1=tf(p,q);%calculo de la función de transferencia analógica en el dominio de s
» [b,a]=butter(n, fd(1)/fn); %frecuencias normalizadas
» g2=tf(b,a,T);%calculo de la transformada Z
» [H,F]=freqz(b,a,2048,1/T);
» plot(F,abs(H))
» plot(F,angle(H))
```



(a) **(b)**

Figura 1.17 Respuesta en Frecuencia del filtro del Ejemplo 1.4 **a)** Magnitud del filtro Butterworth **b)** Fase del filtro Butterworth

En general todas las técnicas de diseño de filtros estudiadas hasta aquí intentan limitar la respuesta de un filtro digital dentro de las zonas demarcadas en la Figura 1.18, la diferencia entre cada una de ellas radica en los diferentes métodos matemáticos que utilizan para cumplir esas condiciones. Además se observan las desviaciones del filtro diseñado con respecto a la respuesta de un filtro pasa bajas ideal.

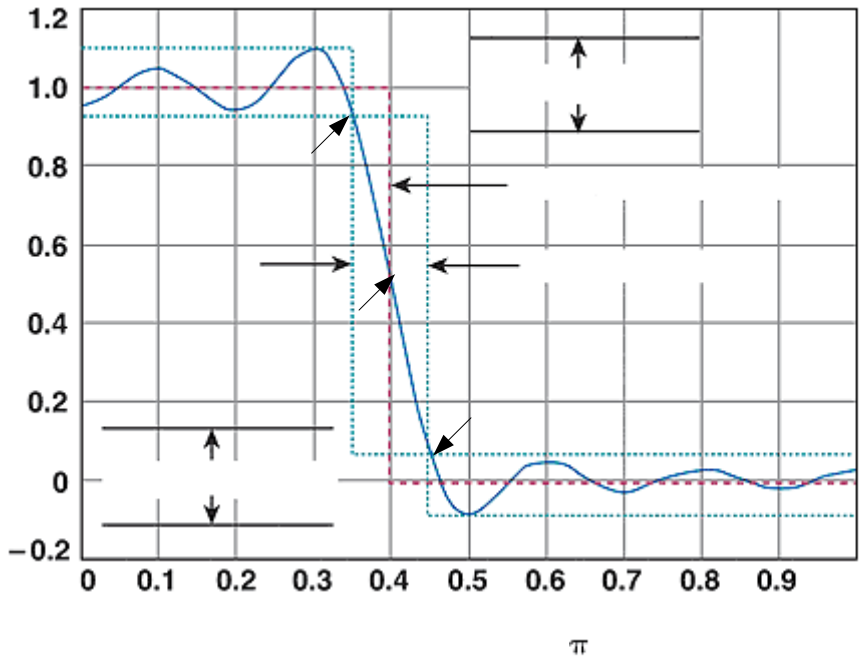


Figura 1.18 Desviaciones típicas de la respuesta de un filtro digital con respecto a un filtro ideal pasa bajas

Ganancia en Wp

Magnitud (V/V)

Ganancia en Wc

2. GENERADOR DE FORMAS DE ONDA ARBITRARIAS

En forma general un generador de formas de onda arbitrarias es un instrumento electrónico capaz de generar una gran variedad de señales analógicas. En particular, el generador implementado en este proyecto de grado es un instrumento virtual programado en LabVIEW que utiliza una tarjeta de adquisición de datos y un computador personal como hardware para generar señales analógicas.

El hecho de ser un instrumento virtual lo convierte en herramienta de laboratorio para el análisis de señales en tiempo y frecuencia, ya que su apariencia y operación imitan a un instrumento físico. Además ya que fue desarrollado en un software dinámico y gráfico como lo es LabVIEW, es posible mejorarlo y adaptarlo a las necesidades del usuario.

El sistema de generación diseñado se basa en la tarjeta de adquisición (TAD) PCI 1200 de National instruments[4], la cual incluye el controlador y el software de aplicación (LabVIEW). El controlador es único para este tipo de dispositivo y se encarga de enviar los comandos que este acepta; el software de aplicación se encarga de enviar comandos al controlador como es preparar la generación o generar por diferentes canales. La ventaja del software de aplicación LabVIEW es que trae una serie de *VIs*[5] integrados con una interfaz gráfica que permiten enviar información a la TAD sin necesidad de escribir extensas líneas de código. Gracias a eso se logró combinar el hardware y el software con tecnologías de computadores personales (PC) estándares para crear un instrumento virtual programado. Gráficamente esto se puede comprender en la Figura 2.1. El primer bloque es el software de aplicación encargado de construir las secuencias de datos con la

[4] Para información técnica sobre la tarjeta de adquisición, referirse al anexo D.

[5] VI, Virtual Instrument. Abreviatura de LabVIEW para referirse a Instrumento Virtual. Veáse más información sobre LabVIEW y sus VIs en el Anexo A.

información de las formas de onda y además de controlar las opciones de generación. Este bloque conecta con el controlador de la TAD enviándole las tareas de generación designadas por el programa. El controlador se encarga de enviar los comandos correctos a la tarjeta de adquisición de manera que se generen de forma correcta los datos enviados por el software. El controlador es programado de fábrica y su código es transparente al usuario. Por último la TAD tiene el hardware adecuado para generación analógica como son conversores D/A y filtros de reconstrucción pasa bajas que junto con los *buffers* almacenados en la memoria del PC generan la señal analógica.

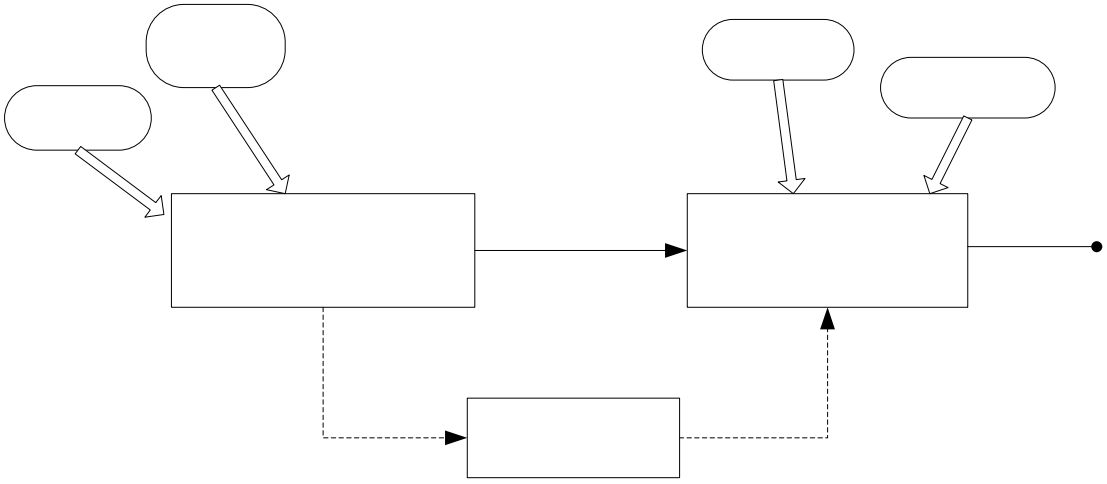


Figura 2.1. Interacción del software y hardware de la TAD.

Este sistema de generación virtual con tarjeta de adquisición se diferencia de los instrumentos de propósito especial, como son los generadores de señales que tienen un *firmware*[6] integrado al hardware que realiza toda la operación de generación y es invisible al usuario; en que su software es dinámico y el hardware está integrado al PC mediante una tarjeta de adquisición conectada al puerto PCI. En otras palabras la TAD y el PC trabajan como un generador de señales. Los componentes del sistema de generación con tarjeta de adquisición se ilustran en la Figura 2.2.

[6] Programa de fábrica escrito en ROM que no puede ser modificado por el usuario.

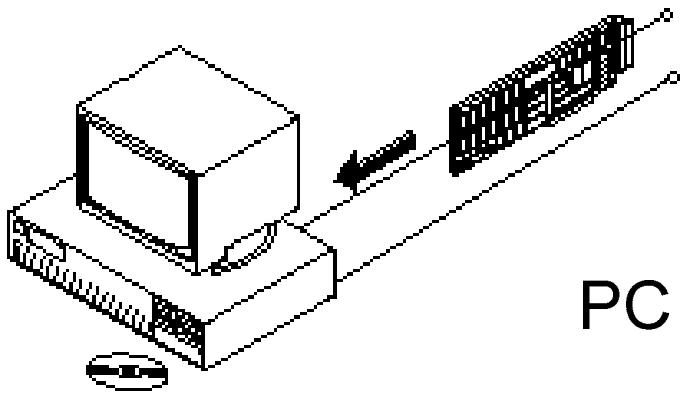


Figura 2.2. Sistema de generación.

La tarjeta de adquisición se encarga solamente de transformar las señales digitales en señales físicas analógicas, por tanto el software del generador es el encargado de realizar previamente todas las tareas necesarias para generar señales reales que el usuario pueda entender como son límites de voltaje, frecuencia y tipo de señal. El software también controla el sistema de la TAD en aspectos como cuando comenzar o detener la generación y en que canales realizar la generación (véase Figura 2.1).

Las condiciones necesarias para poder hacer uso del generador de señales son: Una tarjeta de adquisición compatible con LabVIEW que se encuentre correctamente instalada con su controlador instalado adecuadamente, un computador personal con la tarjeta de adquisición conectada a una ranura PCI y el software de adquisición diseñado.

Software

Un generador de señales arbitrarias tiene una gran ventaja sobre los generadores comunes de señales periódicas básicas. Esta ventaja es la gran cantidad de aplicaciones como son: diseño de señales multitonales de audio, pruebas de dispositivos de análisis cardiaco, simulación de señales de comunicaciones, implementación de señales para estimular circuitos, generación de pulsos, simulaciones en tiempo real, diseño y prueba de señales mezcladas, simulación y mejoramiento de daños en la calidad de señales básicas, importación de formas de ondas de bases datos almacenadas en archivos de texto y generación de señales en tiempo real para evaluar la respuesta en frecuencia de filtros de cualquier tipo.

2.1 ESPECIFICACIONES DEL GENERADOR

El generador tiene las siguientes especificaciones:

2.1.1 Formas de Onda de Salida

- Generación de señales estándares: senoidales, triangulares, cuadradas y sierras.

- Generación de señales por medio de funciones matemáticas: señales AM, funciones sinc, suma de tonos, señales DC, señales exponenciales, señales senoidales amortiguadas, señales rectificadas y cualquier otra señal que se pueda expresar con una fórmula matemática.

- Generación de señales dibujadas con el *mouse*.

- Generación de señales importadas de archivos de texto.

También cuenta con las siguientes opciones para manipular las formas de ondas citadas anteriormente:

  - Adición de señales
  - Diferencia de señales

2.1.2 Resolución de la Forma de Onda

Resolución horizontal (número de muestras): El *buffer* se almacena en la memoria del PC y puede generar señales de hasta 100 kilo muestras.

Resolución vertical: Los dos conversores D/A de la tarjeta PCI 1200 cuentan con 12 bits de resolución vertical con lo cual se obtienen 4096 niveles de voltaje. La TAD puede

generar señales bipolares con amplitudes entre -5V y + 5V. También por medio del software *MAX* [7] de la TAD se puede seleccionar un rango unipolar entre 0 y 10V.

2.1.3 Frecuencia de Salida

Empíricamente la tarjeta PCI 1200 demostró generar con una frecuencia de muestreo de hasta 50kilo-muestras/seg. Teóricamente esta frecuencia de muestreo permitiría una frecuencia pico de generación de 25kHz. En la práctica la máxima frecuencia de salida estable recomendada es 1kHz. Durante la generación simultánea LabVIEW escribe un dato de salida por cada canal durante cada intervalo de muestreo.

2.1.4 Modos de Operación

El generador puede operar en dos modos: Generación continua y generación finita. En modo continuo la señal de salida está presente continuamente en el canal de salida hasta que el usuario lo permita. En modo finito la señal de salida está presente en el canal de salida un tiempo establecido por el usuario.

2.1.5 Canales de Salida

El generador es de doble canal simultáneo. Se puede generar por dos canales, el canal 0 y/o el canal 1 los cuales corresponden al pin 10 y pin 12 respectivamente de la tarjeta PCI 1200 con tierra análoga en el pin 11. Estos pines están disponibles de la bornera integrada con la tarjeta de adquisición[8]. La máxima corriente que puede entregar el generador a una carga es de 2 mA.

Este generador se diseñó para crear una amplia gama de señales de prueba para evaluar el diseño de filtros digitales FIR e IIR. No es adecuado para aplicaciones que requieran

---

[7] MAX, *Measure & Automation.* Software integrado con la TAD que permite configurar diferentes modos de operación.

[8] Para más información sobre la configuración de los pines de la TAD, referirse al anexo D.

generar señales de alta frecuencia (del orden de 100kHz o 1MHz), con excelente calidad en su respuesta en frecuencia y una gran estabilidad. Esto se debe a que la TAD PCI 1200 de National Instruments no está diseñada para este tipo de aplicaciones pues es una tarjeta de bajo costo. Sin embargo para evaluar el diseño de filtros digitales se presenta como una potente herramienta, ya que permite diseñar y generar señales con una información de frecuencia que sea modificada con la aplicación de un filtro específico.

2.2 PANEL FRONTAL DEL GENERADOR DE SEÑALES ARBITRARIAS

El generador de señales arbitrarias cuenta con una interfaz gráfica programada en LabVIEW que permite editar fácilmente una gran gama de formas de onda para ser generadas analógicamente.

El panel frontal del generador de señales puede ser manipulado por el usuario mediante el *mouse* del PC y fue diseñado para tener una apariencia similar a la de los generadores de formas de ondas básicas.

Sin embargo es recomendable que el usuario adopte una metodología para la edición de las formas de onda para que su experiencia con el generador sea eficiente y así obtener los resultados esperados.

En la Figura 2.3 se observa el panel frontal del generador de señales arbitrarias.

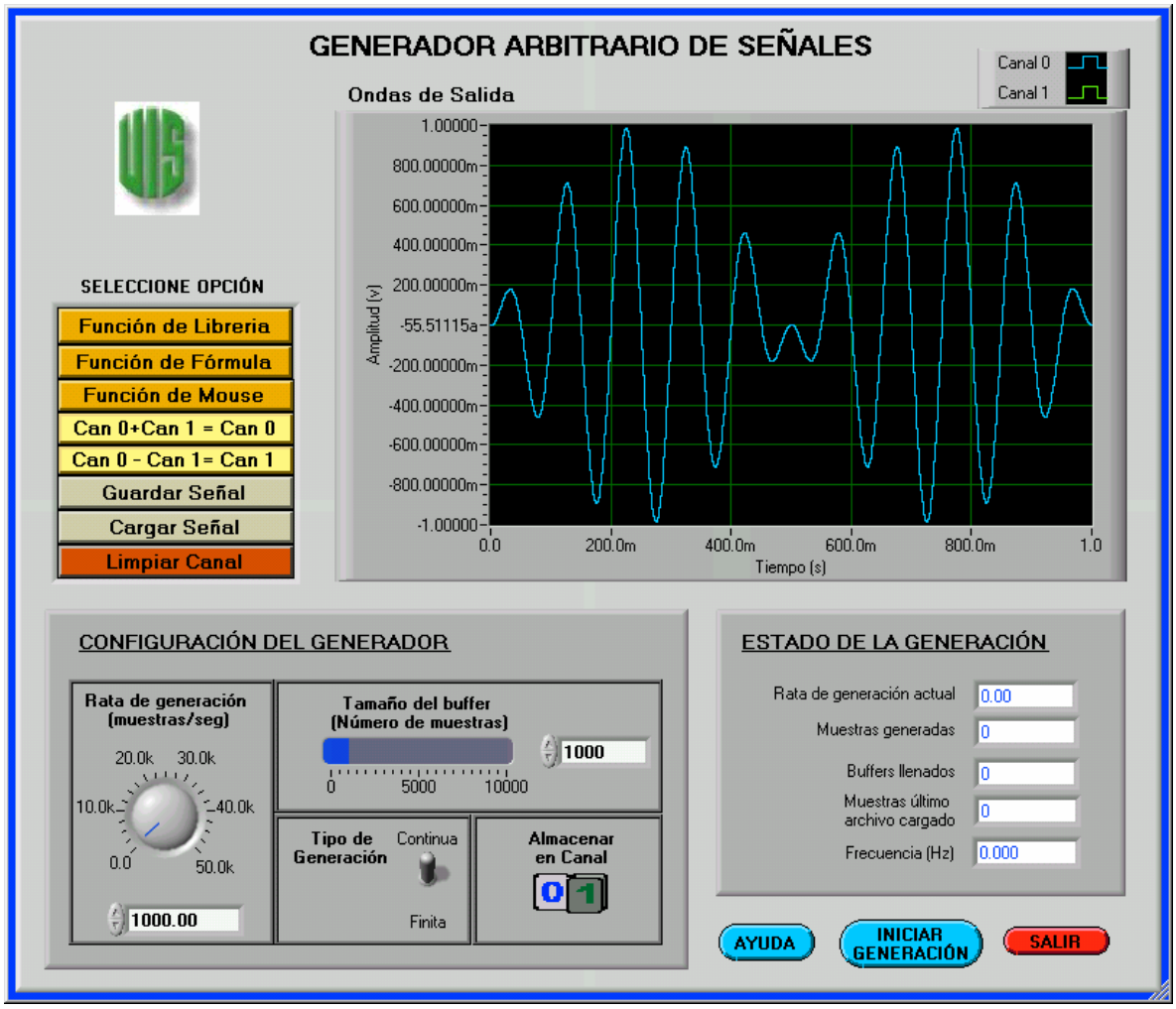


Figura 2.3. Panel Frontal del Generador de Señales Arbitrarias

En el panel frontal del generador de señales arbitrarias se pueden visualizar cuatros zonas principales: 1- Configuración del generador, 2- Selección de opción, 3- Ondas de salida y 4- Estado de la generación. En el Anexo *B* se encuentra la guía del usuario para utilizar correctamente el generador de señales arbitrarias. El panel frontal se despliega al ejecutar el archivo *Generador Arbitrario.VI*

A continuación se explicará detalladamente el algoritmo en LabVIEW que lleva a cabo todas las tareas que se necesitan para generar señales analógicas con el generador de señales arbitrarias.

2.3 GENERACIÓN DE FORMAS DE ONDA

Al generar formas de onda se puede pensar en dos tipos de señales, señales constantes y señales que varían con el tiempo. LabVIEW tiene subprogramas especiales para este tipo de señales.

2.3.1 Generación Analógica Sencilla

Cuando el nivel de la señal es más importante que la velocidad a la cual cambia la señal con el tiempo, se necesita generar un nivel de DC constante. Si se desea este tipo de señal se utilizan *VIs* que generan punto a punto. En la Figura 2.4 se observa el algoritmo en LabVIEW para generar un valor DC constante.

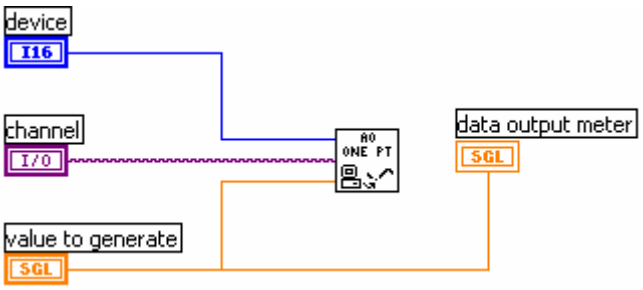


Figura 2.4 Generación Analógica Sencilla utilizando *AO one pt.VI*

2.3.2 Generación Analógica con *Buffer*

En algunas aplicaciones es importante tener en cuenta tanto el nivel de la señal como la rata de generación necesaria para producir la señal con la frecuencia deseada. Una forma eficiente de lograr esto es la generación con *buffer*. Para lograr esto se crea un arreglo de datos correspondientes a la señal deseada por el usuario para llenar el *buffer*, estos datos se pasan al algoritmo de generación y luego el *buffer* se genera continuamente a una rata de generación establecida por el usuario. Gracias a esto no se sacrifica la eficiencia del sistema ejecutando comandos de configuración, localización y liberación del *buffer* cada

vez que se ejecuta el algoritmo de generación, ya que esto produce demoras de tiempo entre cada *buffer* generado.

En la Figura 2.5 se muestra el diagrama de bloques simplificado para una generación analógica con *buffer*.

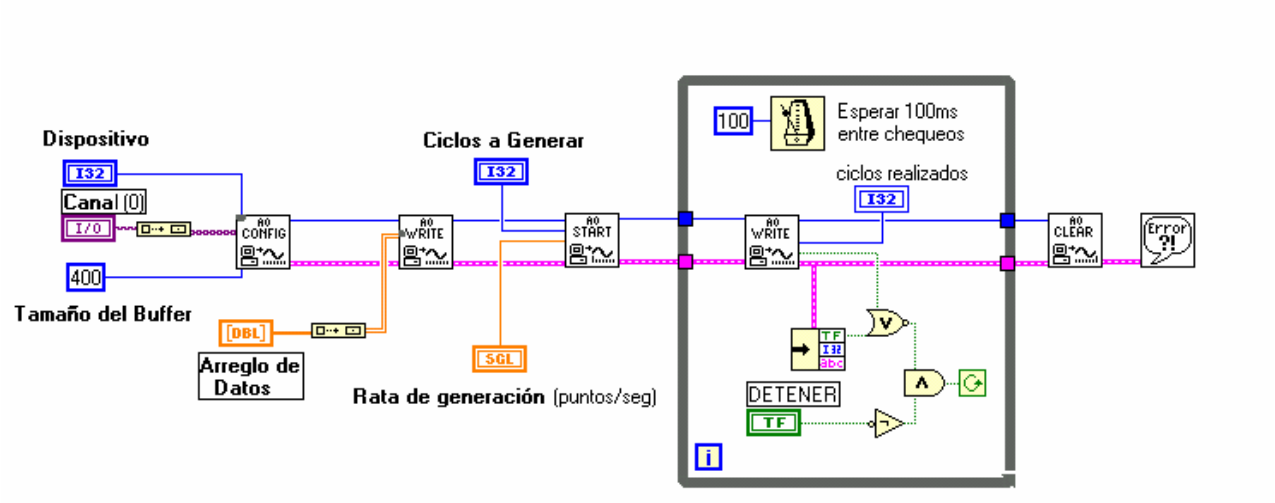


Figura 2.5 Programación de la generación analógica con *buffer* utilizando *VIs* intermedios.

Este diagrama utiliza *VIs* de salida análoga intermedios que permiten establecer varios parámetros de generación. *AO Config.VI* configura la tarjeta de adquisición localizando el tamaño del *buffer*, el canal de salida y el dispositivo. *AO Write.VI* prepara los datos a ser generados. *AO Start.VI* da comienzo a la generación y permite establecer la rata de generación y el número de veces que será generado el *buffer* antes de terminar la generación. El ciclo *WHILE* permite detener la generación en cualquier momento, chequea si surge algún error e indica cuantos *buffer* se han generado. Finalmente *AO Clear.VI* limpia la salida al terminar la generación liberando el *buffer* y se muestra en mensaje de error en caso de que lo haya. Este tipo de generación es el indicado para la aplicación de generación y por tanto se explicarán con detalle los *VIs* utilizados durante esta generación.

2.3.3 Descripción de los *VIs* Utilizados en la Generación con *Buffer*

A continuación se describen las terminales de entrada y salida de los *VIs* utilizados en la generación con *buffer* para comprender su funcionamiento y cuales comandos envían a la TAD. Estos *VIs* son de nivel intermedio y son los más recomendados para realizar una programación flexible y utilizar las potencialidades de la TAD. Las entradas en negrilla son las más comúnmente utilizadas y deben ser cableadas como requerimiento para que el programa se pueda ejecutar; las entradas claras son opcionales y no necesitan ser cableadas para la ejecución del programa. Los valores entre paréntesis () son los valores por defecto de las entradas. Los valores en *brackets* ([ ]) son raramente utilizados.

2.3.3.1 AO Config.vi

Este *VI* configura los canales, los límites de la señal generada y el tamaño del *buffer*. Se observa en la Figura 2.6.

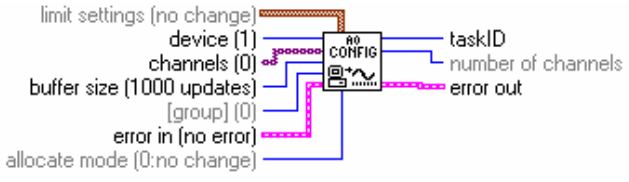


Figura 2.6 *AO Config. VI.* Tomada de *LabView Help*

- *Limit settings:* Son los límites de señal para cada canal. Para el generador de señales se tomaron los valores de voltaje por defecto de la tarjeta PCI 1200 que son -5 V a 5 V.

-Device: Es el número del dispositivo de adquisición que está asignado por el software al instalar la TAD. La tarjeta PCI 1200 de NI tiene asignado el número uno (1).

-Channels: Es una lista de los canales de salida análogos que se desean utilizar, editados en una cadena de texto. Si x, y, z son canales, se puede especificar una lista de canales separando cada canal individual por comas en una cadena de texto (por ejemplo, x,y,z). Si x es el primer canal en un rango consecutivo de canales y z es el último canal, se puede especificar el rango de canales separando el primer y último canal por dos

puntos en una cadena de texto (por ejemplo, x:z). La tarjeta PCI 1200 tiene dos canales de salida análoga configurados como cero (0) y uno (1).

-Buffer size: Es el número de muestras que se desea que el *buffer* almacene.

-Error in: Describe las condiciones de error que se presentaron antes de la ejecución del VI. Si hay un error el valor de Error in pasa a Error out.

1) Status: Dato de tipo Booleano. Es verdadero si ha ocurrido un error, y por tanto el *VI* no ejecuta ninguna operación.

2) Code: Es el número del código de error que identifica al error ocurrido.

3) Source: Identifica el VI donde ocurrió el error. El tipo de dato es un *string* con el nombre del VI fuente del error.

-Allocate mode: Indica a LabVIEW como designar la memoria para el *buffer*. El software diseñado utilizará el valor por defecto de esta entrada (0) que designará la memoria del PC para el *buffer*.

-Task ID: Identifica la operación de E/S.

-Error out: Contiene información de error. Si *Error in* indica un error *Error out* tendrá exactamente la misma información. De lo contrario Error out tendrá la información del error ocurrido en el *VI*.

2.3.3.2 AO Write.vi

Este VI Escribe datos en el *buffer* para generar la señal análoga. Puede manejar dos tipos de datos: arreglos de datos sin información de tiempo o formas de onda con todos sus atributos. Se observa en la Figura 2.7.

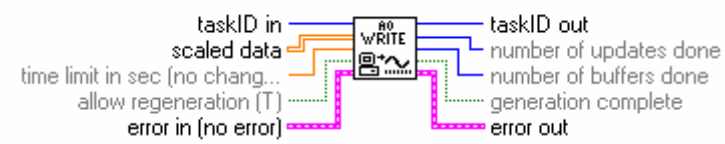


Figura 2.7 *AO Write.VI.* Tomada de *LabView Help*.

-***Task ID in:*** Esta entrada trae la información de la tarea de generación que se está realizando. Por ejemplo trae datos de operación provenientes de *Task ID out* de *AO Config.Vi.*

–***Scaled data:*** Es una matriz de 2 dimensiones que contiene la información de la señal que se desea generar en la unidades físicas correspondientes. Para el generador de señales arbitrario las señales son de voltaje. La matriz está ordenada de manera que cada columna de datos corresponda a un canal y cada fila tiene el correspondiente dato de cada canal.

-Time limit in sec: Esta opción permite establecer el tiempo de duración de la generación del *buffer*. Terminado este tiempo se puede rescribir información en *buffer* si este no se ha generado en su totalidad. Para el caso del generador arbitrario la entrada tiene su valor por defecto el cual permite a LabVIEW establecer el tiempo para que la forma de onda pueda ser generada en su totalidad antes de escribir nuevos datos en el *buffer*.

-***Allow generation:*** Para el generador arbitrario de señales esta entrada debe estar en su valor por defecto el cual permite generar el *buffer* repetidas veces, sin necesidad de escribir información nueva en él. De esta forma cada vez que LabVIEW termina de generar el *buffer* en su totalidad regresa de nuevo al principio del *buffer* y lo vuelve a generar sin importar si la información ha sido o no actualizada, o sea se repiten los datos anteriores de la generación en caso no que no se hayan introducido unos datos nuevos.

-***Error In:*** Esta entrada contiene la misma información de error explicada en el anterior VI. En caso de que Error In indique información de un error, el VI no se ejecuta y la información se pasa a Error out.
-Task ID out: Esta salida tiene el mismo valor que *Task ID in.*

-Number of updates done: Es el número de veces que el VI ha generado un dato. Esto lo mismo que el número de muestras que el VI ha transferido desde el *buffer* hasta el conversor D/A de la TAD para ser generadas.

-***Number of buffers done*:** Es el número de veces que se han generado todas las muestras del *buffer*. Esto es lo mismo que el número de veces que el VI ha transferido el *buffer* completo al conversor D/A de la TAD.

-Generation complete: Esta salida tiene un valor verdadero cuando la generación se ha completado. Es de tipo Booleano.

-***Error out:*** Contiene la información de error.

2.3.3.3 AO Start.vi

Este VI da inicio a la generación análoga con *buffer*. Establece un parámetro de gran importancia como lo es la rata de generación. Se observa en la Figura 2.8.

Figura 2.8 *AO Start. VI.* Tomada de *LabView Help.*

-TaskID in: Identifica cual es la tarea de Entrada/Salida por realizar.

***-Number of buffer iterations*:** Este número indica a LabVIEW cuantas veces debe generar la forma de onda almacenada en el *buffer* de salida. Por defecto es uno (1) lo que resultaría en la generación de la información del *buffer* una sola vez y luego se detendría la generación. Para una generación continua del *buffer* esta entrada se debe establecer

en cero (0) con la cual LabVIEW genera los datos del *buffer* de salida continuamente y la información del *buffer* continuará en la salida de la TAD. Para detener la operación de generación y liberar el *buffer* se tiene que ejecutar el *AO Clear. VI.*

-Update rate: Es el número de muestras del *buffer* generadas por segundo. Esta entrada permite controlar la frecuencia o rata de generación.

-Error In: Describe las condiciones de error antes de que se ejecute el VI. Si la entrada indica algún error el VI no se ejecuta y retorna el valor de error in en error out.

-TaskID out: Esta salida tiene el mismo valor que la entrada *TaskID in.*

-Error out: Esta salida contiene información de error.

2.3.3.4 AO Wait.vi

Este VI espera hasta que la tarea de generación de la forma de onda termine antes de continuar el flujo de datos. Se usa para esperar que una generación con *buffer* finita termine antes de llamar a *AO Clear.VI.* Se observa en la Figura 2.9.

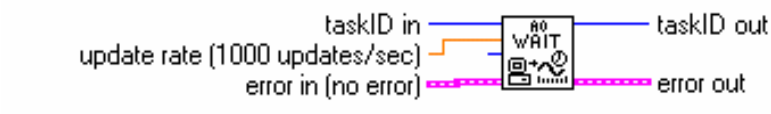


Figura 2.9 *AO Wait. VI.* Tomada de *LabView Help.*

-TaskID in: Identifica la tarea de entrada. Para el caso del generador arbitrario su tarea de entrada es una generación con *buffer* finita.

-Update Rate: Es la rata de generación. El número de muestras a generar por segundo. Es el mismo dato utilizado en *AO Start.VI* para configurar la generación.

-Error in/out: Contiene información de error.

-***TaskID out:*** Contiene la misma información que *TaskID in*.

2.3.3.5 AO Clear.vi

Este *VI* limpia el canal de salida análoga asociado a su tarea de entrada (TaskID in). Sin importar si ha existido un error o no, este VI detiene la generación y libera recursos internos como *buffers*. Se observa en la Figura 2.10.

Figura 2.10 *AO Clear. VI.* Tomada de *LabView Help.*

-TaskID in: Identifica la tarea de entrada. Es decir el canal de salida análogo que se desea limpiar.

-Error in/out: Contiene información de error. Pero sin importar si indican o no error, este VI se ejecuta. Si ha ocurrido un error durante la operación de generación esta información es de utilidad para visualizar diálogos de error.

-TaskID out: Esta salida tiene el mismo valor de TaskID in.

2.4 ESTRUCTURA DEL ALGORITMO DEL GENERADOR DE SEÑALES

Gracias a la programación por flujo de datos en LabVIEW, el algoritmo de programación del generador de señales se divide en tres bloques principales que se ejecutan en un orden secuencial.

- BLOQUE 1-CONSTRUIR LAS ONDAS: Construir las formas de onda en arreglos de datos unidimensionales de acuerdo a los parámetros introducidos por el usuario.

- BLOQUE 2-PREPARAR LOS DATOS: Chequear la información de los arreglos de datos y prepararla para su almacenamiento en el *buffer*. Además realiza una previsualización de la onda de salida agregando la información de tiempo correcta a los arreglos de datos construidos en el primer bloque.

- BLOQUE 3-GENERACIÓN ANALÓGICA CON *BUFFER*: Escribir la información en el *buffer* y comenzar la generación análoga por medio de la TAD PCI 1200. Luego de terminada la generación se liberan los recursos internos del PC como *buffers* y se establece la salida en cero.

Gráficamente lo explicado anteriormente se observa en la Figura 2.11.

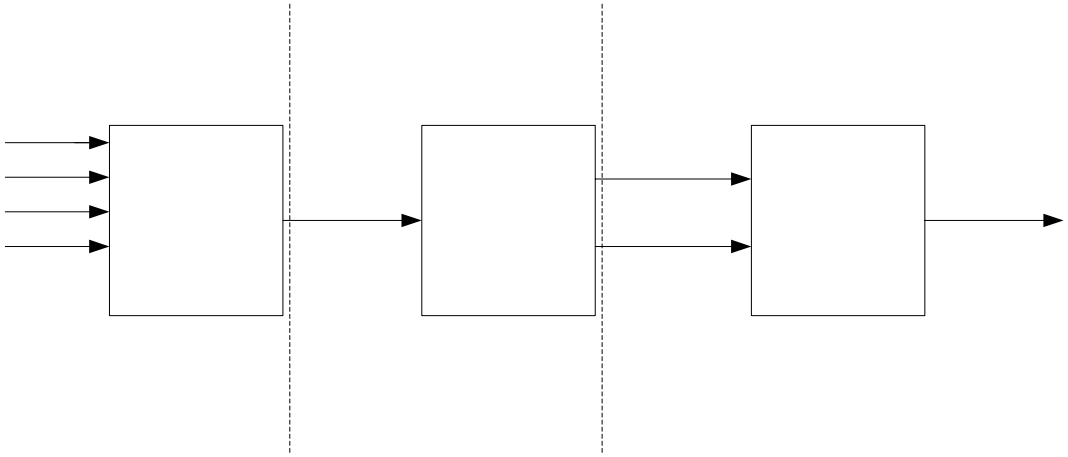


Figura 2.11 Diagrama de bloques del algoritmo de generación.

Se explicará ahora detalladamente cada bloque del algoritmo.

2.4.1 Bloque 1- Construir las Ondas

El algoritmo comienza esperando por la selección del usuario. El usuario puede seleccionar entre ocho casos diferentes almacenados en un *cluster* compuesto de ocho controles booleanos correspondientes a cada caso. Los controles se establecen en modo mecánico de pulsador y se realiza la búsqueda de un valor verdadero dentro de los componentes del *cluster* gracias a un conversor de *cluster* a arreglo y la aplicación *Search 1D Array*[9]. Esto se puede observar gráficamente en la Figura 2.12.

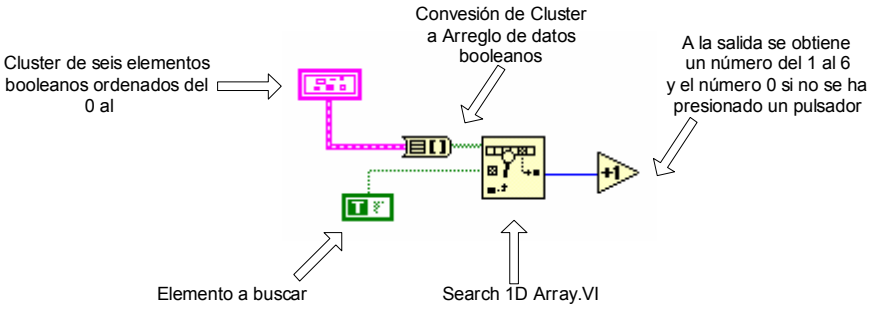


Figura 2.12 Esquema para la selección de opción.

Esta aplicación tiene la función de buscar dentro del arreglo de datos booleanos constantemente la pulsación de uno de los controles para ejecutar el caso correspondiente. El incremento de 1 se utiliza para crear un caso "*stand by*" cuando el usuario no ha presionado ningún pulsador. Los casos asignados a cada pulsador en su orden dentro del *cluster* son como se observa en primera columna de la Tabla 2.1. La segunda columna de la tabla corresponde a los nombres de los textos booleanos de los controles y la tercera columna corresponde al número de salida de la aplicación luego del incremento. Se observa que el caso 0 nunca puede ser seleccionado por el usuario y por tanto es un caso de espera de selección o "stand by" del programa. En la Figura 2.13. se observa el menú de selección de la opción tal y como aparece en el panel frontal del generador de señales arbitrarias.

[9] Para una descripción detallada de los instrumentos virtuales utilizados a lo largo del texto referirse al Anexo A.

Tabla 2.1. Números asignados para cada caso seleccionado por el usuario

No. Ent.	Opción	No. Salida
0	Función de Librería	1
1	Función desde Mouse	2
2	Función de Fórmula	3
3	Can 0 + Can 1 = Can 0	4
4	Can 0 + Can 1 = Can 1	5
5	Guardar Señal	6
6	Cargar Señal	7
7	Limpiar Señal	8

Figura 2.13. Menú de selección de opción

Luego de seleccionada la opción el algoritmo ejecuta el caso correspondiente.

2.4.1.1 Función de Librería

Es el caso número 1. Se ejecuta al presionar la opción Función de Librería. En la Figura 2.14 se observan las entradas y salida de este caso. Se detalla también la forma como la aplicación de búsqueda de dato booleano selecciona el caso correspondiente. También se ilustra una estructura de secuencia dentro del *case* con dos acciones secuénciales. En la figura se observa la acción cero (0) donde se ejecuta el subprograma[10] *Librería*. La acción número uno (1) no se muestra pero en ella se guarda la correspondiente frecuencia estimada de salida de la señal que se construirá con el subprograma.

[10] Referirse al anexo A, para entender como maneja LabVIEW los subprogramas.

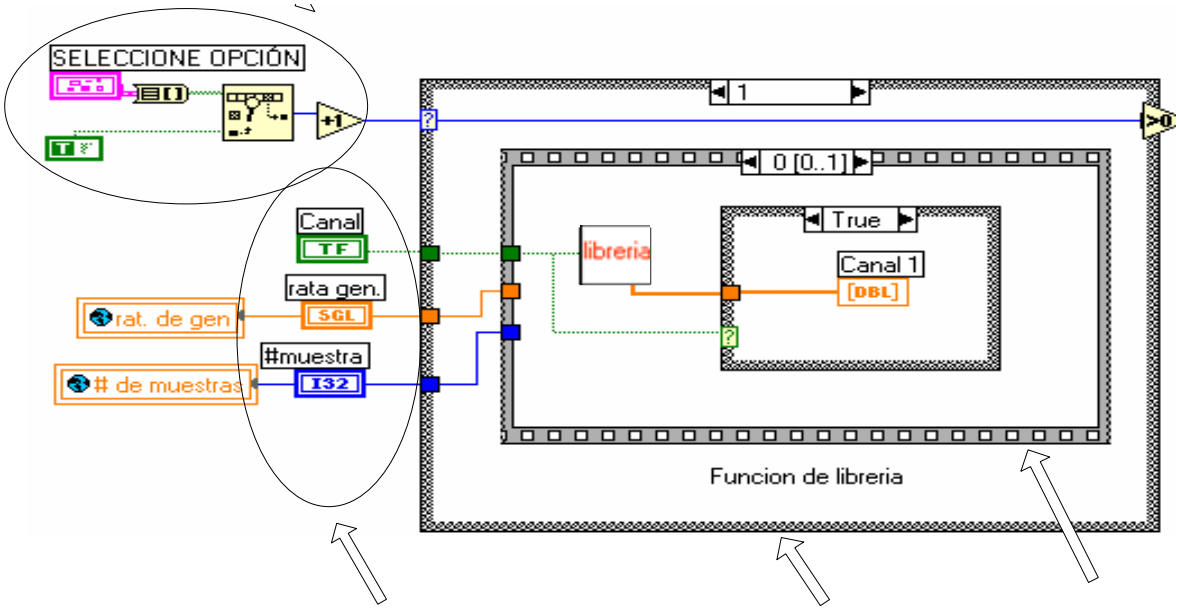


Figura 2.14 Caso Función de librería.

Al entrar a este caso se ejecuta el subprograma *Librería*. Este subprograma tiene como entradas el canal de salida análogo, la rata de generación del *buffer* y el número de muestras del *buffer*. Como salidas tiene el vector de datos correspondientes a la señal editada. Mediante un caso *True-False* se escoge donde se guardará el vector de datos, si en el canal 1 o en el canal 0.

El panel frontal del subprograma *Librería* se muestra en la Figura 2.15. En esta figura se observa una gráfica con información de tiempo donde se visualiza la señal que se va a generar. El color de la señal indica por cual canal de salida analógica se generaría la señal en caso de que el usuario inicie la generación. El color azul indica el canal 0 y el color verde indica el canal 1. Para editar la señal se utilizan los controles que se observan en la Figura 2.15. Estos controles son: tipo de señal, el cual se puede escoger entre señal senoidal, triangular, cuadrada o sierra; amplitud en volts, la cual debe estar dentro de los límites de voltaje permitidos por la TAD; frecuencia en hertz de la señal, fase en grados, *offset* o nivel de DC y ciclo útil[11]. En la figura se observa como ejemplo un

Entradas de configuración del generador

[11] Opción disponible solo para señal cuadrada.

período de una señal senoidal de 1 volt de amplitud ,1Hz de frecuencia y sin *offset* ni desfase que se podría generar por el canal 0.

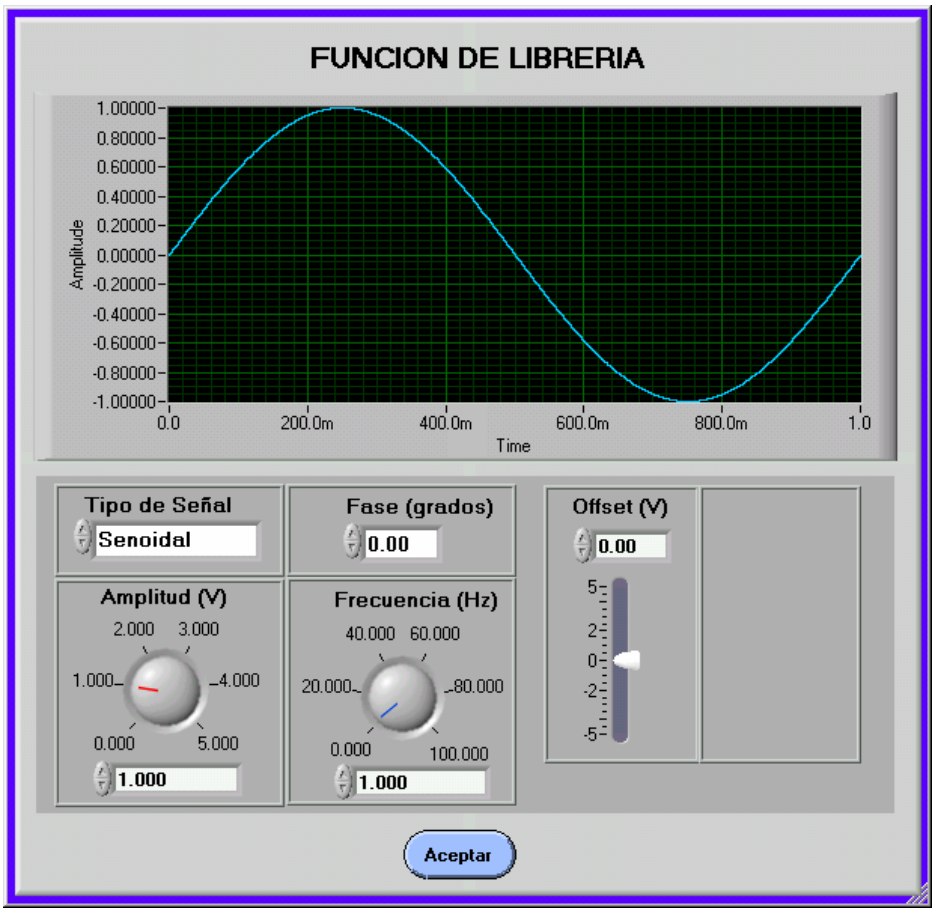


Figura 2.15 Panel frontal del subprograma Librería.

El algoritmo del subprograma tiene como aplicación central a *Basic Function Generador.VI* el cual se observa en la Figura 2.16. Este VI crea una forma de onda de salida común como son: senoidales, triangulares, sierras o cuadradas. Tiene como entradas el tipo de señal, el número de muestras, la fase y la frecuencia de la señal.

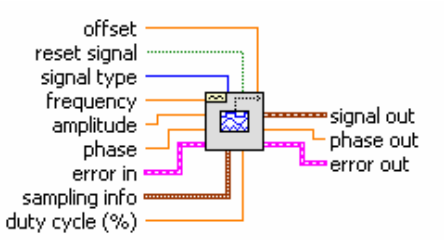


Figura 2.16 *Basic Function Generador.VI.* Tomada de *LabVIEW Help.*

Los datos de amplitud, *offset*, tipo de señal, frecuencia, amplitud, fase y ciclo útil; son ingresados por el usuario desde el panel frontal del subprograma y son las entradas de la aplicación. Las entradas de información de muestreo como son la rata de generación y el número de muestras son introducidas a la aplicación utilizando variables globales que tienen asignadas los datos de configuración del generador que se ingresaron por el usuario desde el panel frontal principal del generador (véase Figura 2.3). Esto debe ser así para que exista congruencia entre la información de tiempo del arreglo de datos construido por la aplicación y la señal que vamos a generar. A la salida de la aplicación se obtiene un *cluster* que contiene un arreglo de datos con la información de tiempo extraída de las entradas rata de generación, número de muestras y frecuencia. Para facilitar el procesamiento de los datos de la forma de onda, se extrae del *cluster* únicamente el arreglo de datos para procesarlo con simples operaciones entre vectores de datos. En la Figura 2.17 se muestra parte del algoritmo en detalle.

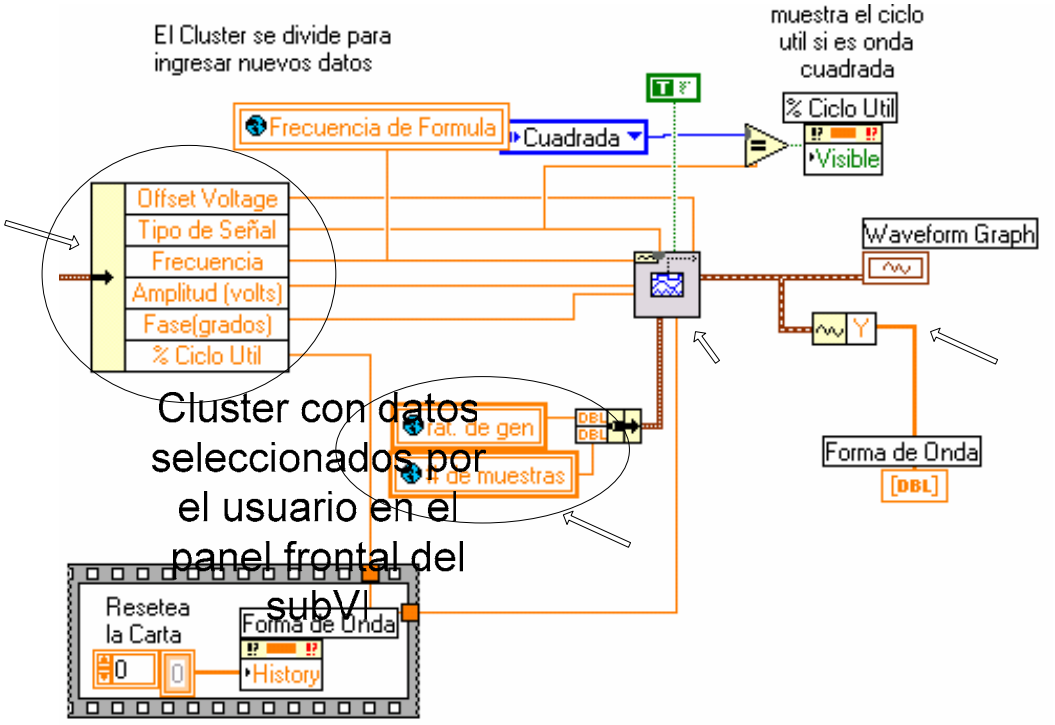


Figura 2.17 Detalle del diagrama de bloques del subprograma *Librería.*

2.4.1.2 Función de Fórmula

Es el caso número 2. Se ejecuta al presionar el pulsador de Función de Fórmula. Como se observa en la Figura 2.18 al entrar a este caso se ejecuta el subprograma *Editar Form*. Este subprograma tiene como entradas el canal de generación análogo, la rata de generación y el número de muestras, las cuales son las mismas del caso 1 y ya se han mostrado en la Figura 2.14. Como salida tiene el vector de datos correspondientes a la señal editada. Mediante un caso *True-False* se escoge donde se guardará el vector de datos, si en el canal 1 o en el canal 0. Dentro del caso 2 se observa una estructura de secuencia en la acción cero (0) donde se ejecuta el subprograma *Editar Form*. La acción número uno (1) no se muestra pero en ella se guarda la correspondiente frecuencia estimada de salida de la señal que se construirá con el subprograma.

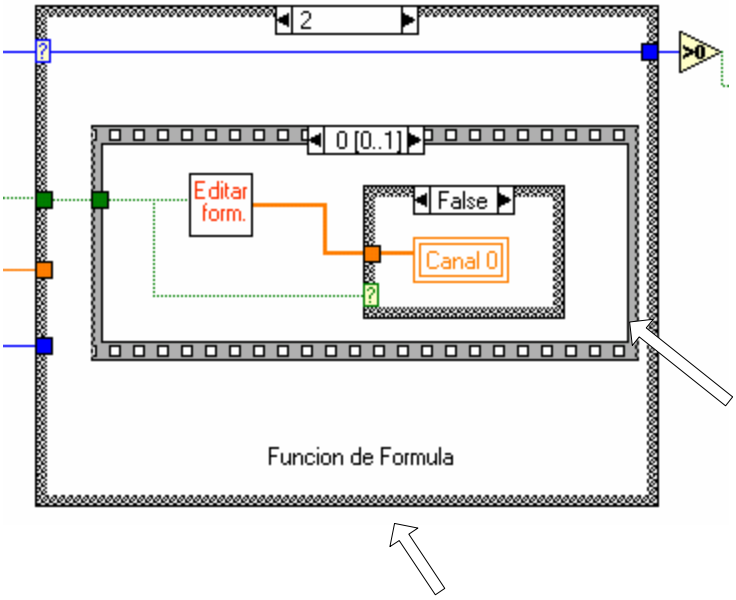


Figura 2.18 Detalle del diagrama de bloques del subprograma *Editar form.*

El panel frontal del subprograma *Editar Form* se muestra en la Figura 2.20. En esta figura se observa una gráfica con información de tiempo donde se visualiza la señal que se va a generar. Al igual que en el subprograma *Librería* se toma la misma convención de color en las señales para indicar su canal de salida. El color azul indica el canal 0 y el color verde indica el canal 1. El algoritmo del subprograma *Editar form* tiene como aplicación principal a *Formula Waveform* el cual se observa en la Figura 2.19. Este VI crea una forma de onda de salida a partir de una fórmula ingresada en su *string* de entrada.

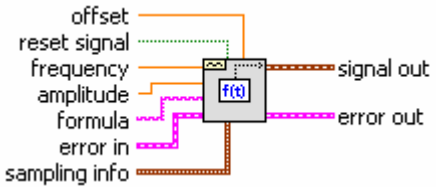


Figura 2.19 *Formula Waveform.VI*. Tomada de *LabVIEW Help.*

Para editar la señal se utilizan los controles que se muestran en la Figura 2.20. Para ingresar la fórmula hay una casilla en blanco donde se puede ingresar la fórmula mediante el teclado. La amplitud en volts se representa mediante la variable “a” en la

casilla de fórmula y se varía con el control de amplitud. La frecuencia en hertz se representa mediante la variable “f” (donde w=2*pi*f) y se varía también son su correspondiente control de frecuencia. También se puede variar el *offset* de la señal.

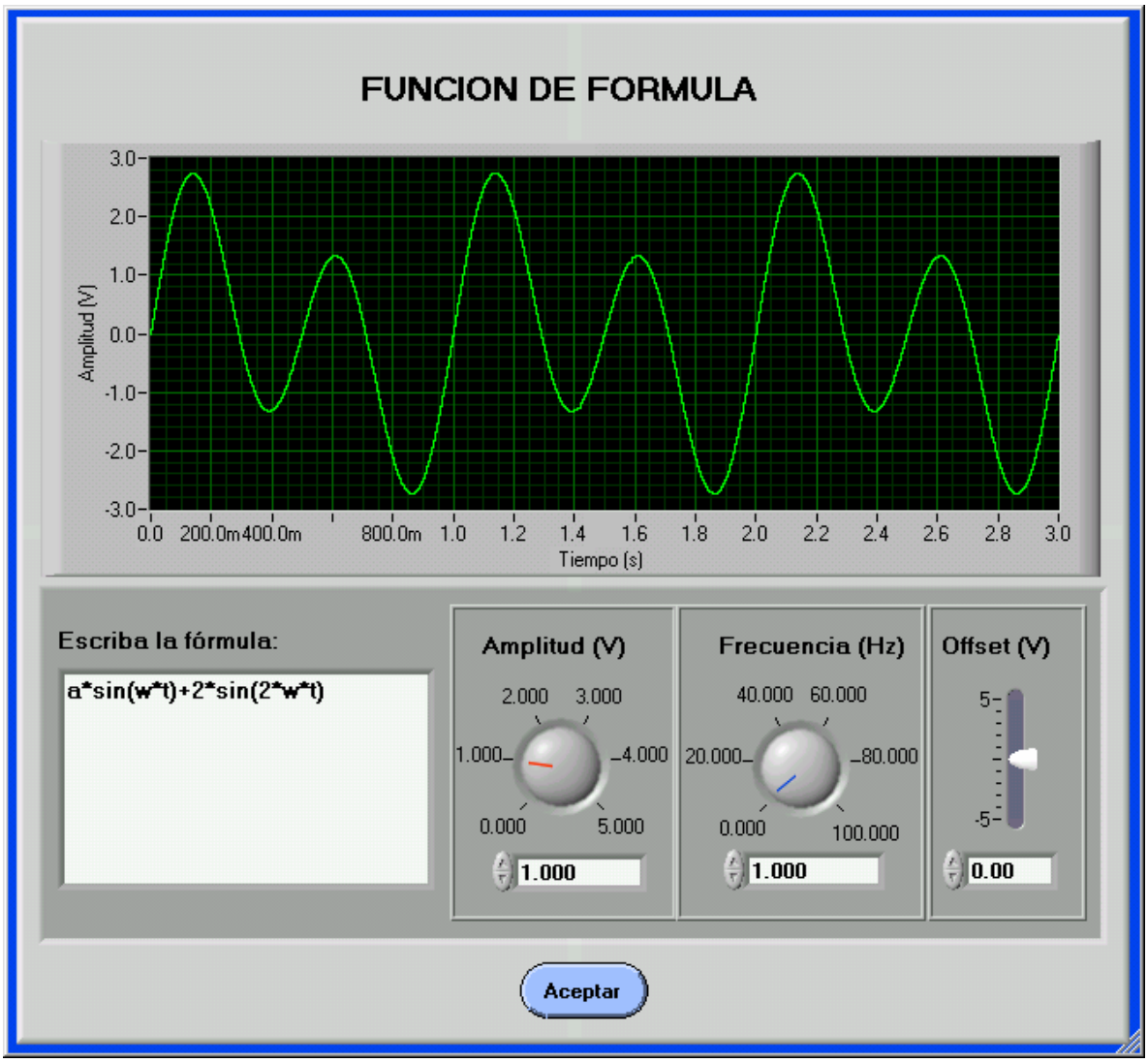


Figura 2.20. Panel frontal del subprograma *Editar Form.*

En la Figura 2.20 se observa como ejemplo tres períodos de la fórmula: y(t)=a*sin(w*t)+2*a*sin(2*w*t), la cual reemplazando los valores de los controles correspondientes a las variables “a” y “w” representa la señal sen(2*pi*t)+2*sen(4*pi*t) [V] que se podría generar por el canal 1.

Como se observa en la Figura 2.21 la aplicación tiene como entrada un *cluster* con los datos de los controles ubicados en el panel frontal los cuales tienen que separarse del *cluster* antes de entrar a la aplicación. Esta organización a manera de *cluster* permite un mejor manejo del algoritmo y además mediante la lógica que se observa en la figura y un caso *True-False* permite optimizar la ejecución de la aplicación únicamente cuando existen cambios en los controles de entrada. La información de muestreo de la aplicación se ingresa mediante variables globales que tienen almacenados los valores de configuración del generador ingresados por el usuario desde el panel frontal del generador de señales arbitrarias (Véase Figura 2.3).

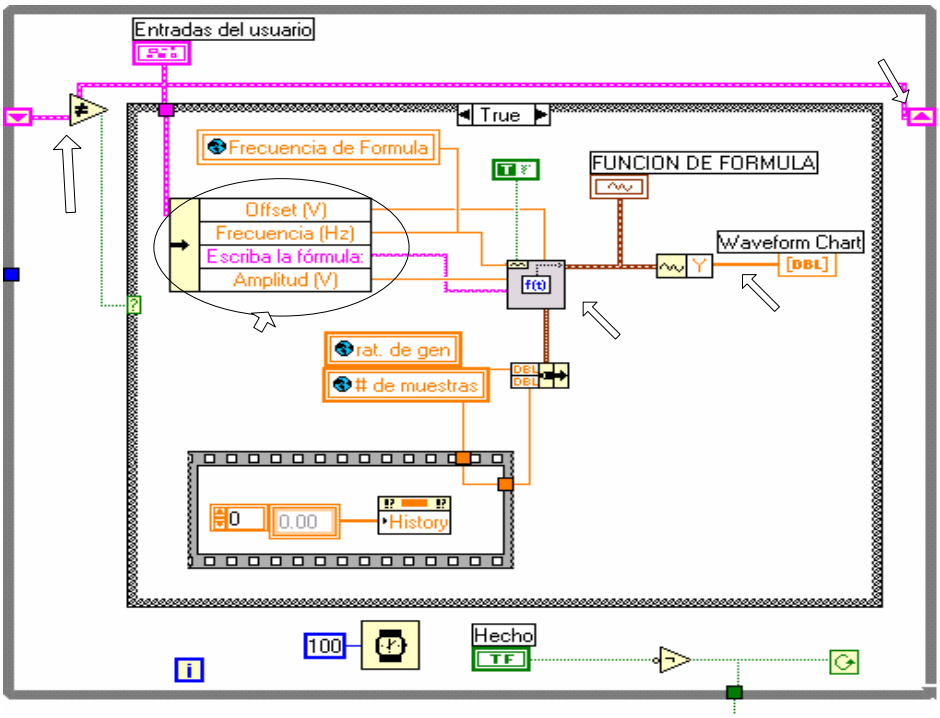


Figura 2.21 Detalle del diagrama de bloques del subprograma *Editar Form*.

2.4.1.3 Función de Mouse

Es el caso número 3. Se ejecuta al presionar el pulsador Función desde Mouse. Al entrar a este caso se ejecuta el subprograma *Mouse*. En la Figura 2.22 se observa como este subprograma tiene como entrada el canal de generación análoga y como salida el vector de datos correspondiente a la señal graficada con el *mouse*. La salida del vector está

comprueba si
hay cambios en
los datos
de entrada

acotada a 1000 muestras por la aplicación *Array Subset.VI*; lo cual significa que la gráfica de *mouse* tendrá un máximo de 1000 puntos. Mediante un caso *True-False* se escoge donde se guardará el vector de datos, si en el canal 1 o en el canal 0. También se observa una estructura de secuencia en la acción cero (0) donde se ejecuta el subprograma y se almacena el arreglo de datos. La acción uno (1) no se muestra en la figura pero en ella se guarda la frecuencia de salida para este caso. En el caso de Función de Mouse la frecuencia de salida está determinada únicamente por la rata de generación establecida en la configuración del generador ya que un período de la señal siempre serán 1000 muestras.

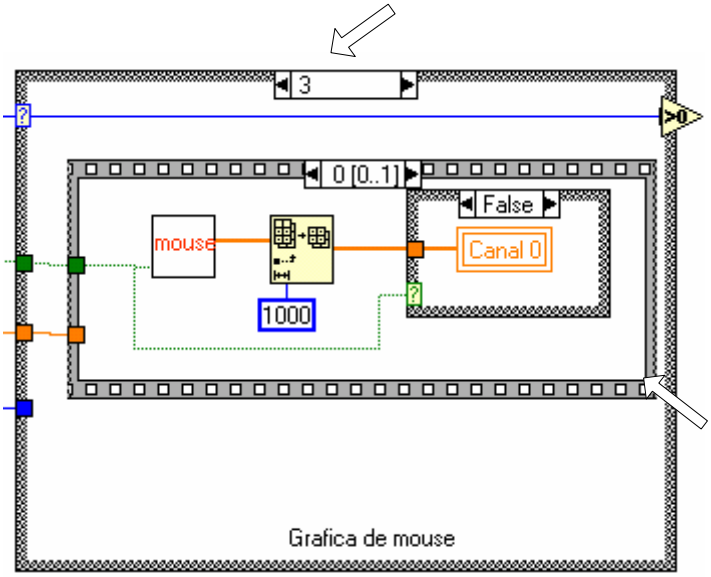


Figura 2.22 Caso Función de Mouse.

Se grafican con el *mouse* únicamente funciones de tipo y=f(x) y se debe tener en cuenta que los límites de salida de la tarjeta de adquisición son de -5V a 5V para señales bipolares o de 0V a 10V para señales unipolares. Además como siempre son 1000 muestras la duración de un período de la señal se podrá calcular como

$$\frac{1000(muestras)}{rata.de.generación(\frac{muestras}{segundo})} \qquad (2.1)$$

y la frecuencia de salida al generar la señal continuamente será por tanto de aproximadamente

$$0.001 * rata.de.generación(Hz) \quad (2.2)$$

El panel frontal del subprograma *Mouse* se ilustra en la Figura 2.23. Se observa que se ha dibujado una señal bipolar con una amplitud de 4 volts aproximadamente. La frecuencia de salida de esta señal generada continuamente se calcula como se explicó anteriormente. En el panel frontal del programa *Generador Arbitrario* se indica la frecuencia estimada para esta señal. El cursor blanco es la guía para dibujar la señal y se puede pensar en él como la punta de un lápiz con el cual se dibujará la señal. Su movimiento se controla naturalmente con el *mouse* o ratón del PC.

Figura 2.23 Panel Frontal del subprograma *Mouse.*

El algoritmo de este subprograma es una modificación del subprograma de LabVIEW *arbitrary waveform sketchpad.VI* el cual esta leyendo constantemente las coordenadas en

“x” y “y” de los cursores mediante nodos de propiedades que tienen almacenada la posición de los cursores en cualquier momento. Si hay algún cambio en la posición inicial del cursor que se establece por defecto el subprograma procede a dibujar un nuevo punto y a interpolarlo linealmente. De esta forma se dará la impresión al usuario de que esta graficando una forma de onda continua cuando en realidad se trata de un arreglo de datos de 1000 muestras equiespaciadas.

2.4.1.4 Adición de Señales en el Canal 0

Es el caso número 4. Se ejecuta al presionar el pulsador CH 0 + CH 1 = CH 0. Como se observa en la Figura 2.24 al ejecutarse este caso el arreglo de datos almacenado en la variable local en modo de lectura del canal 0 se suma con el arreglo de datos almacenado en la variable local en modo de lectura del canal 1 y este resultado se sobrescribe en el arreglo de datos de la variable local en modo de escritura del canal 0. Es recomendable sumar señales con el mismo número de muestras para no tener problemas con el tamaño del *buffer* ya que este podría quedar con espacios vacío si una señal tiene más muestras que otra y por tanto el espacio sin muestras de la señal más corta no está definido y no se realiza la suma provocando espacios sin muestras en el *buffer* de salida y una frecuencia de salida errónea.

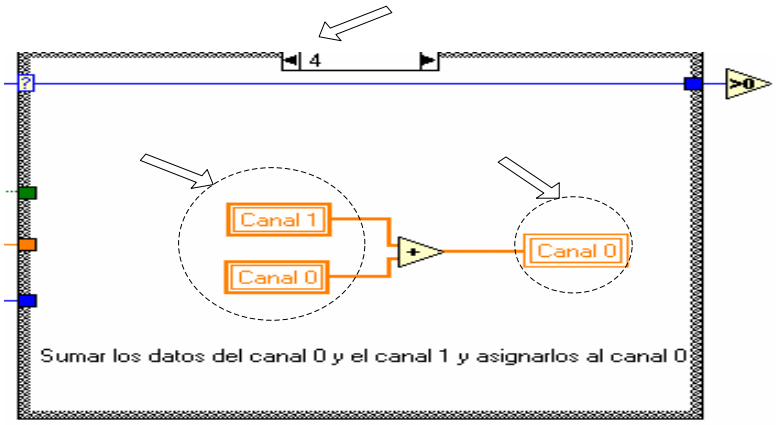


Figura 2.24 Caso de adición de señales.

2.4.1.5 Diferencia de señales en el Canal 1

Es el caso número 5. Se ejecuta al presionarse el pulsador CH 0 - CH 1 = CH 1. Como se observa en la Figura 2.25 al ejecutarse este caso el arreglo de datos almacenado en la variable local en modo de lectura del canal 0 se restan con el arreglo de datos almacenado en la variable local en modo de lectura del canal 1 y este resultado se sobrescribe en el arreglo de datos de la variable local en modo de escritura del canal 1. Para este caso se aplican las mismas recomendaciones del caso anterior sobre el número de muestras de cada señal.

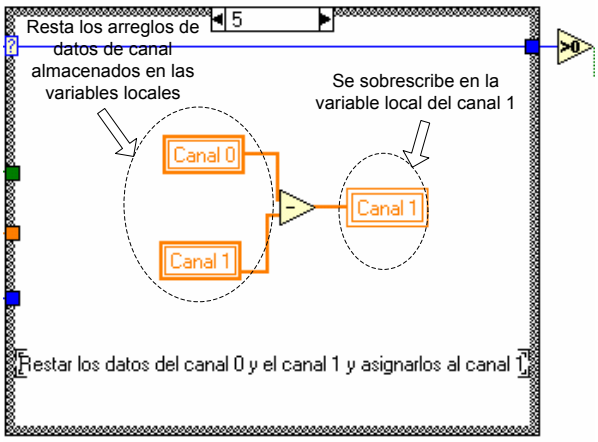


Figura 2.25 Caso diferencia de señales.

2.4.1.6 Guardar Señal

Es el caso número 6. Se ejecuta al presionar el pulsador Guardar Señal. Como se observa en la Figura 2.26 al ejecutarse este caso los datos del canal 0 o el canal 1 son escogidos mediante un caso *True-False* para ser guardados en un archivo de texto gracias a la aplicación *Write to Spreadsheet File*. Esta aplicación se configura para transponer el arreglo de datos de forma que queden organizados en una columna dentro del archivo de texto. Es decir que cada dato tenga un retorno de carro dentro del archivo de texto.

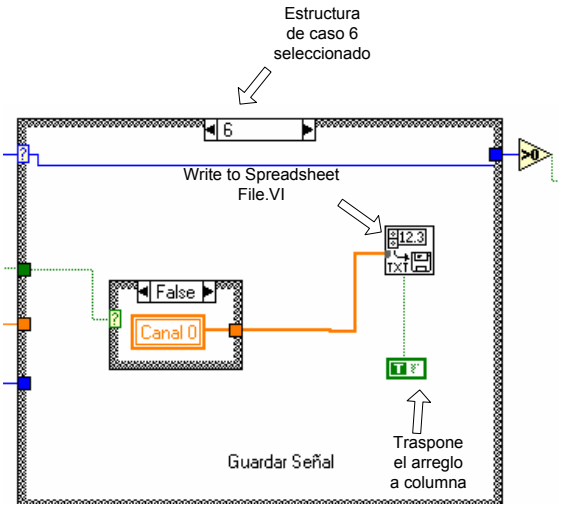


Figura 2.26 Caso guardar señal.

2.4.1.7 Cargar Señal

Es el caso número 7. Se ejecuta al presionar el pulsador Cargar Señal. Como se observa en la Figura 2.27 al ejecutarse este caso los datos del canal 0 o el canal 1 son escogidos mediante un caso *True-False* para ser cargados de un archivo de texto gracias a la aplicación *Read to Spreadsheet File*. Esta aplicación se configura para transponer los datos del arreglo de datos de forma que queden organizados a manera de fila dentro del arreglo.

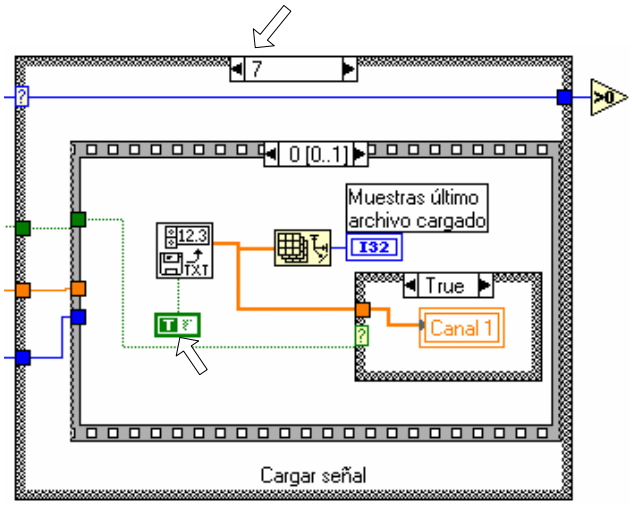


Figura 2.27 Caso cargar señal.

2.4.1.8 Limpiar Canal

Es el caso número 8. Se ejecuta al presionar el pulsador Limpiar Canal. Como se observa en la Figura 2.28 al ejecutarse este caso se escribe un vector de ceros al canal 1 o al canal 0 dependiendo del canal que se desee limpiar. Esta dependencia se programa con un caso *True-False*.

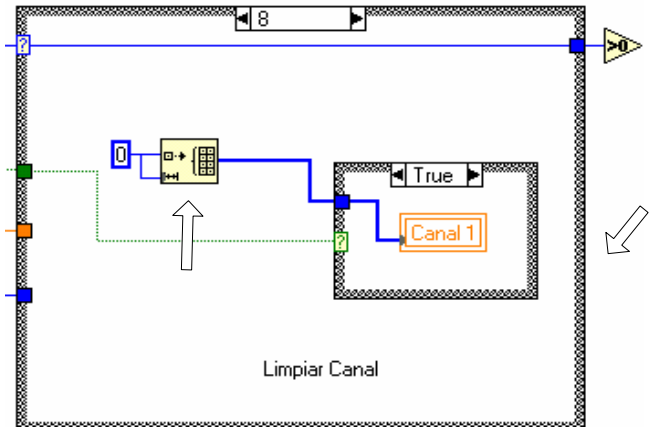


Figura 2.28 Caso limpiar canal.

2.4.2 Bloque 2- Preparar los Datos

Es un bloque de enlace de gran importancia entre el primer y el tercer bloque. Este bloque se encarga de preparar los arreglos de datos que contienen las formas de ondas construidas por el primer bloque de manera que el tercer bloque los pueda entender.

Este bloque contiene una serie de casos anidados que dependen de la selección del usuario y de cómo se debe organizar la información de los arreglos de datos de las formas de onda.

El caso de mayor nivel es de tipo *True-False* y divide el algoritmo en dos casos: cuando el usuario ha seleccionado alguna opción y otro cuando el algoritmo está en modo "stand by" en espera de la selección de una opción.

Crea un vector de
de ceros

2.4.2.1 Caso *FALSE*

Este caso se ejecuta cuando no se ha seleccionado ninguna opción. Está configurado de forma tal que en el evento que el usuario presione el control iniciar generación ubicado en el panel frontal del generador de señales arbitrarias (véase Figura 2.3), una compuerta AND de tres entradas chequea si los canales están vacíos o tienen información para generar. Si están vacíos se visualiza un mensaje indicando que se debe ingresar una forma de onda primero. Si hay datos el algoritmo continua el flujo de datos para generar la información existente. En la Figura 2.29 se muestra en detalle el algoritmo. Se observa que las variables locales que almacenan las formas de onda de cada canal están cableadas a la aplicación *size(s)* la cual indica si los vectores de datos están o no vacíos. Si alguno de los canales contiene información o ambos contienen información la generación puede continuar. Se observa que la única forma de seleccionar el caso *True* de error es si ambos canales están vacíos y se desea iniciar la generación, y por tanto tendríamos tres entradas verdaderas en la compuerta AND y se selecciona el caso *True* anidado que despliega el mensaje de error en el panel frontal del generador de señales arbitrarias.

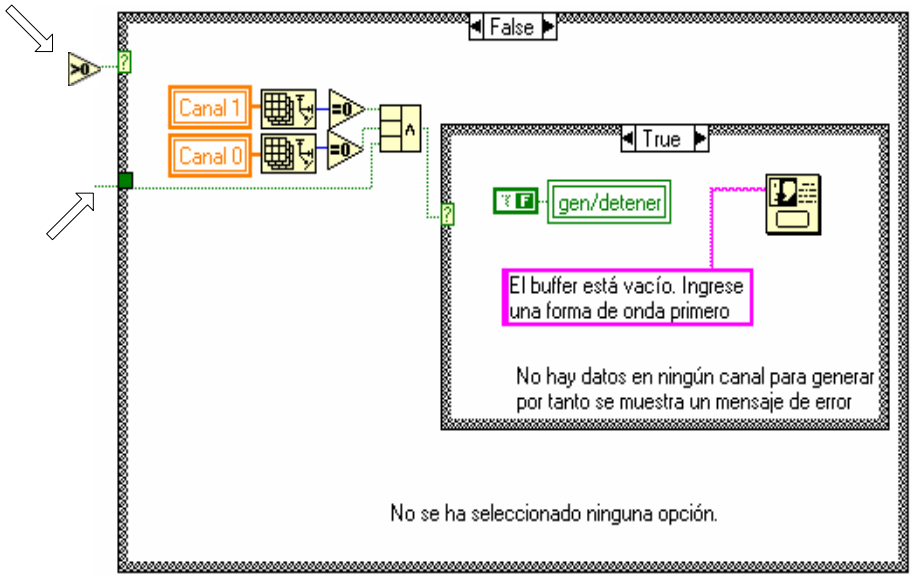


Figura 2.29 Caso *False* del bloque 2.

2.4.2.2 Caso *TRUE*

Este caso se ejecuta cuando el usuario ha seleccionado alguna opción. Dentro de este caso la información proveniente del bloque 1 se organiza dependiendo el caso que ha seleccionado el usuario. Si el usuario ha construido una forma de onda para generar en el canal 0 o en el canal 1, se ejecuta un caso anidado en el cual se configura el arreglo de salida como un vector fila y el canal de salida para enviarlos al bloque 3. En caso de que ambos canales contengan información para generar, se ejecuta un caso en que el arreglo de salida es un matriz de dos dimensiones en donde la primera fila corresponde a los datos del canal 0 y la segunda fila corresponde a los datos del canal 1. Además se configura el *string* de canal para enviarlo al bloque 3 de manera que este genere simultáneamente por el canal 0 y canal 1. La matriz de salida bidimensional es acotada al número de muestras por *buffer* configurado por el usuario. Por último los datos del canal 0 y/o el canal 1 que van a ser generados se les adiciona su información de tiempo independiente seleccionada por el usuario de tal forma que puedan visualizarse las ondas en una gráfica antes de ser generadas. En la Figura 2.30 se puede observar el algoritmo en detalle.

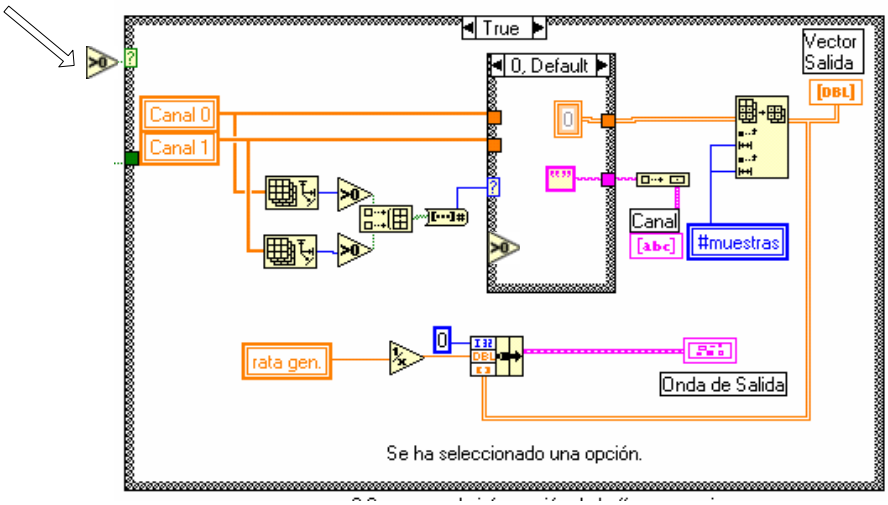


Figura 2.30 Caso TRUE del bloque 2.

2.4.3 Bloque 3- Generación Analógica con *Buffer*

Este bloque es el encargado de enviar los *comandos* correctos a la TAD PCI 1200 y transferir los datos del *buffer* al conversor D/A para iniciar la generación. Este bloque se ejecuta si hay información lista para ser generada y el usuario desea comenzar la generación analógica. Sus datos de entrada son:

2.4.3.1 Tipo de Generación

Selecciona entre una generación continua y una generación finita. Este control se cablea directamente a la aplicación *AO Start*. En modo de generación continua el *buffer* de salida se repetirá consecutivamente hasta que ocurra un error o el usuario desee detener la generación. La opción de generación finita se maneja con la aplicación *AO Wait*. Gracias a esta aplicación se logra detener el flujo de datos hasta terminar la generación finita y de esta forma permitir a la aplicación *AO Clear* establecer las salidas en cero volts al terminar la generación finita.

2.4.3.2 Vector de Salida

Es una matriz bidimensional que debe ser transpuesta para ingresarla correctamente a *AO Write.VI*. La matriz debe tener la configuración mostrada en la Figura 2.31.

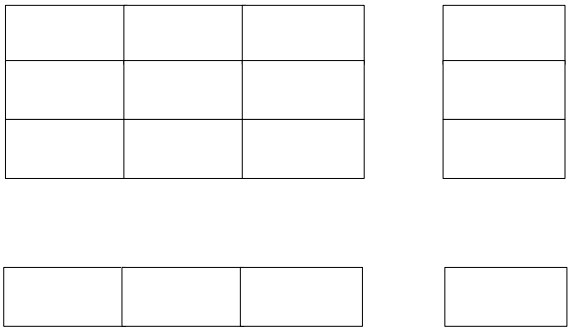


Figura 2.31 Matriz de entrada a *AO Write*

Por ejemplo en el caso de construir un vector de salida en el bloque 2 como el de la matriz A mostrada en la Figura 2.32, la forma correcta de ingresarla a *AO Write* sería luego de trasponerla como se ve en la matriz B. En este caso la TAD PCI 1200 generaría simultáneamente por el canal 0 y el canal 1 las cuatro muestras (en este caso un valor de DC de 1V y 2V respectivamente) a una rata de generación seleccionada por el usuario.

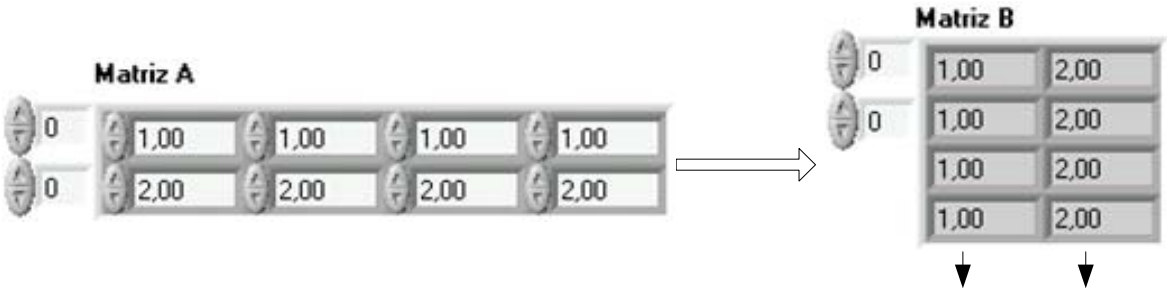


Figura 2.32. Ejemplo de transposición de una matriz.

2.4.3.3 Canal

Es una cadena de texto preparada por el bloque 2 para ingresarla a la aplicación *AO Config* según sea el caso. Los casos son: generar por el canal 0 y/o el canal 1. Cuando se genera por un solo canal (canal 0 o canal 1) la salida del otro canal permanece en 0 V. Cuando se genera simultáneamente se pueden observar en un osciloscopio las dos señales al mismo tiempo. Puede almacenar señales en ambos canales y realizar operaciones de suma o diferencia entre ellos como se explicó anteriormente (véase secciones 2.4.1.4 y 2.4.1.5)

En la Figura 2.33 se muestra en detalle las entradas al bloque 3 y la programación del algoritmo.

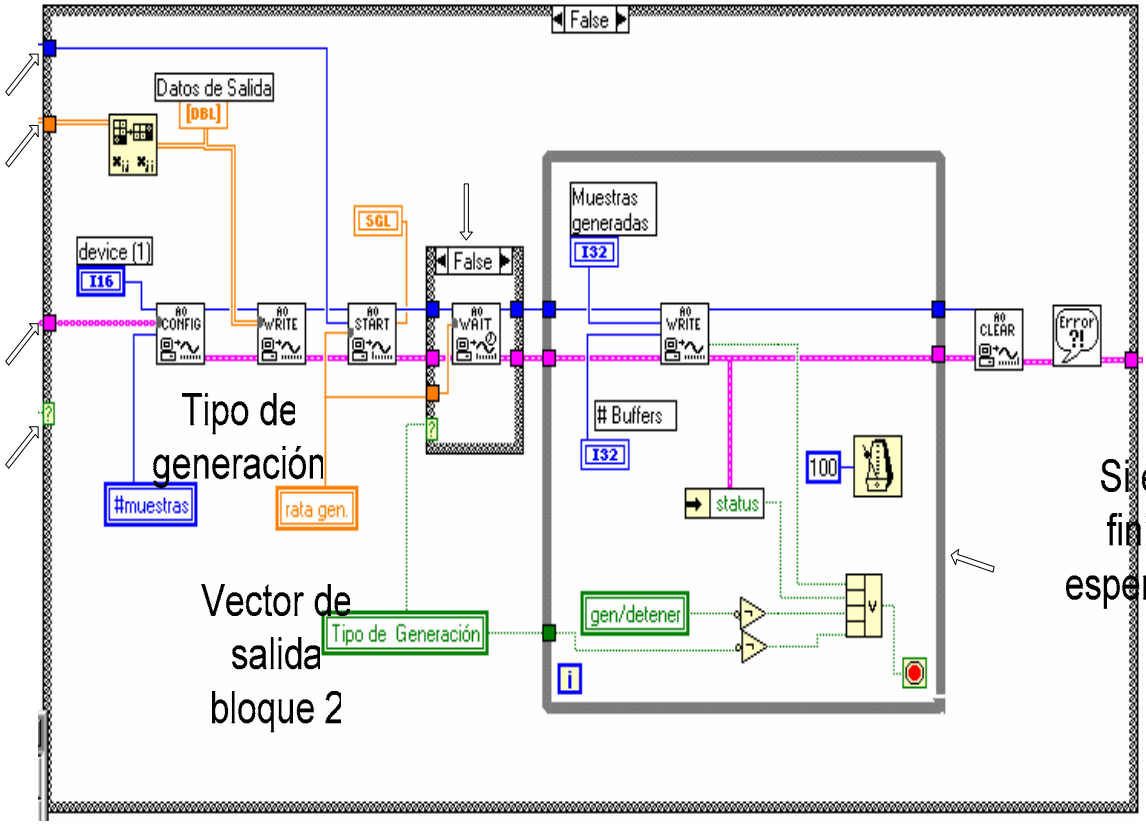


Figura 2.33 Detalle del algoritmo de generación con *buffer* del bloque 3.

Luego de recibidas estas entradas al bloque 3, el algoritmo procede a realizar una generación continua o finita analógica con *buffer*. Durante la ejecución del bloque 3 se obtiene la señal analógica en el canal (o canales) que se ha seleccionado. Esta señal puede entonces ser utilizada para simulaciones, pruebas o cualquier aplicación mencionada anteriormente. La generación continua es terminada cuando el usuario lo desee y presione el control *Detener Generación* en el panel frontal del generador (véase Figura 2.3). Si es generación finita termina automáticamente.

Canal

Se ejecuta si hay datos en el vector de salida

2.5 DIAGRAMAS DE FLUJO DEL ALGORITMO

Dada su ejecución por bloques, a continuación se mostrará un diagrama de flujo de cada bloque para comprender mejor el funcionamiento del algoritmo

2.5.1 Diagrama bloque 1

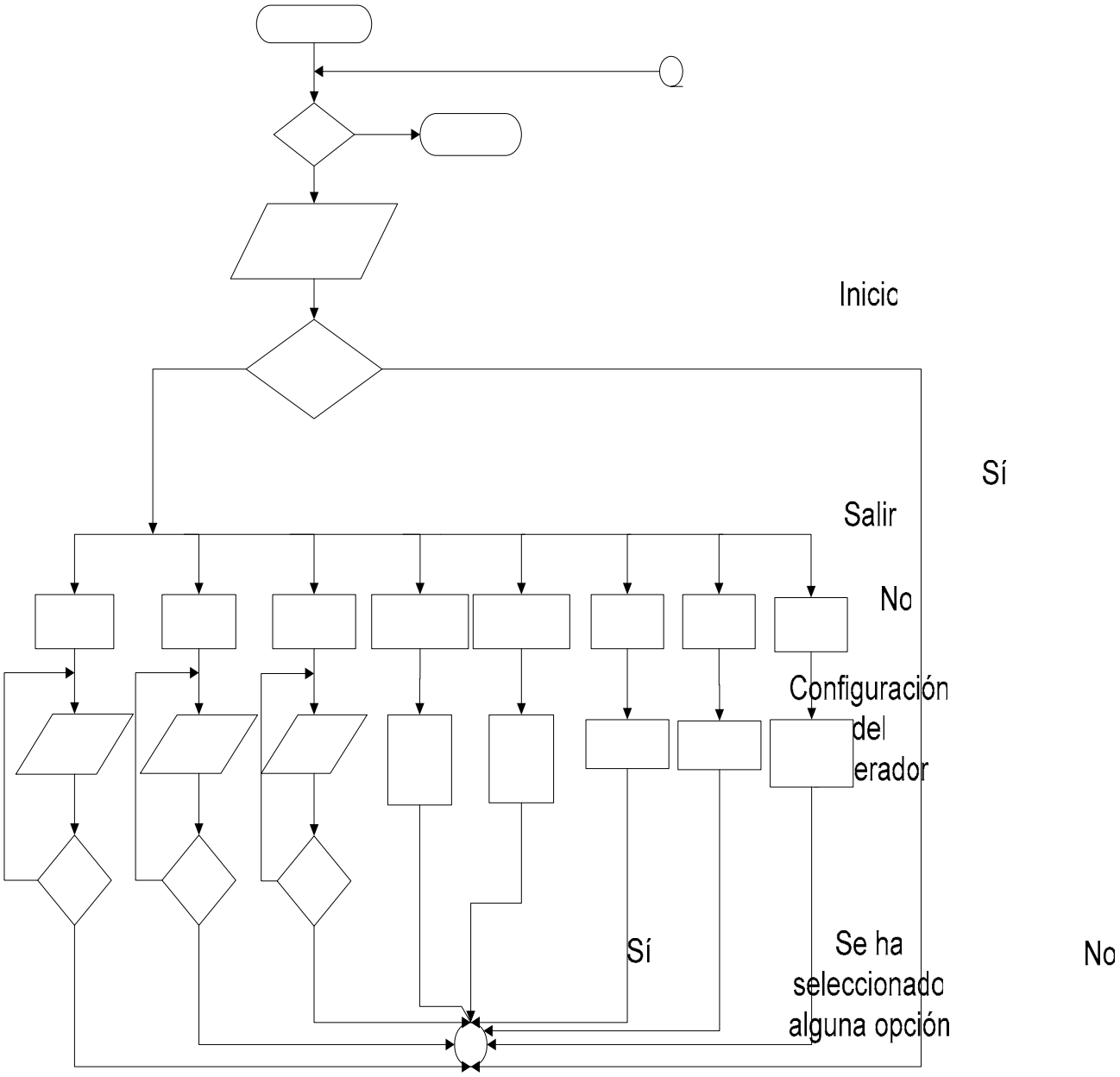


Figura 2.34 Diagrama de flujo del bloque 1.

En la Figura 2.34 se sintetiza la función del Bloque 1, la cual es editar las señales que se desean generar dependiendo de la opción seleccionada por el usuario.

Función de Librería | Función de Fórmula | Función de Mouse | Can 0 + Can 1 = Can 0

2.5.2 Diagrama Bloque 2

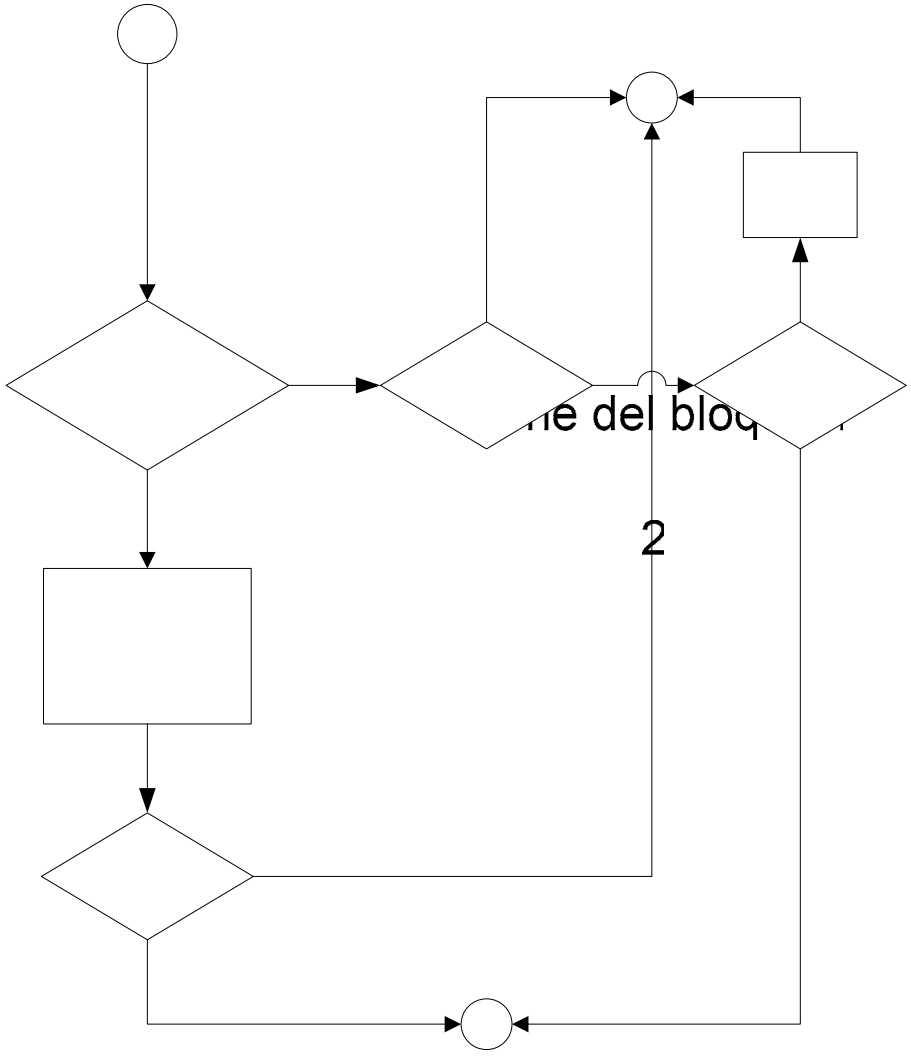


Figura 2.35 Diagrama de flujo del bloque 2.

La función del bloque 2, como se observa en la Figura 2.35, es preparar los datos de salida dependiendo de las opciones de generación configuradas por el usuario.

2.5.3 Diagrama Bloque 3

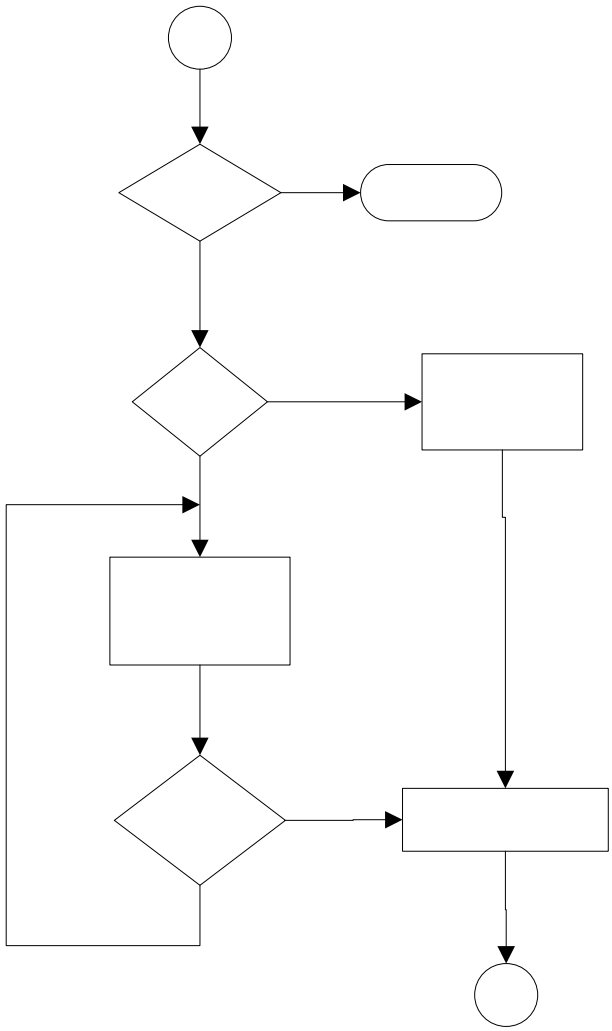


Figura 2.36 Diagrama de flujo del bloque 3.

El bloque 3, como se observa en el diagrama de flujo de la Figura 2.36, se encarga de inicializar la tarjeta de adquisición para una operación de generación analógica con *buffer*. Al finalizar se encarga de liberar los recursos utilizados como *buffers*.

Al terminar de ejecutarse el bloque 3 el algoritmo retorna al bloque 1 a esperar otra selección del usuario y el ciclo se repite constantemente. Al presionar el control de *Ayuda* (véase Figura 2.3.) se despliega una guía corta explicando como editar una forma de onda en pocos pasos. Una guía detallada, con ejemplos de formas de onda editadas con la aplicación, se encuentra en el anexo B. Para finalizar totalmente la ejecución del programa se presiona el control *Salir* en el panel frontal del generador (véase Figura 2.3).

Para una descripción más detallada de todas las aplicaciones de LabVIEW utilizadas referirse al anexo A.

3. ALGORITMOS DE ADQUISICIÓN Y FILTRADO

En el capítulo 1 se mencionaron las técnicas de diseño de filtros digitales recursivos y no recursivos y se resolvieron algunos ejemplos con el fin de ilustrar el procedimiento de diseño de algunos de estos métodos.

Algunos de estos ejemplos se desarrollaron con la ayuda de MATLAB y LabVIEW, sin embargo, muchos de los algoritmos empleados allí, se encuentran integrados en rutinas de LabVIEW para el procesamiento digital de señales.

Este capítulo pretende mostrar las funciones de LabVIEW que realizan estas operaciones y utilizarlas en una aplicación conjunta que permita adquirir una señal y filtrarla en tiempo real.

Para esto se utilizará el método de adquisición *buffereada* y las funciones de filtrado: *Digital IIR Filter.vi, Digital FIR Filter.vi, e IIR Filter.vi* de *LabVIEW National Instruments.*

En primera instancia se explicarán los diferentes métodos de adquisición de señales con el fin de resaltar las ventajas de la técnica *buffereada* y posteriormente se expondrán las funciones de filtrado.

3.1 ADQUISICIÓN DE SEÑALES

LabVIEW es una herramienta muy poderosa en la medición de variables físicas como temperatura, nivel y presión, dado que tiene una enorme capacidad para adquirir señales en tiempo real y procesarlas. Además, LabVIEW puede adquirir una gran variedad de

señales eléctricas, tales como: cardiacas, cerebrales, senoidales, cuadradas, triangulares y aleatorias (que no se pueden expresar como una función matemática).

Al tratar de adquirir señales en LabVIEW, el programador se encuentra con dos formas de hacerlo: por el *buffereado* o por el sencillo. El primero consiste en adquirir muestras de una señal analógica y almacenarlas en un *buffer* localizado en la memoria principal del computador. El segundo sólo permite capturar una muestra a la vez.

Cada uno de estos métodos se aplica dependiendo del tipo de señal que se vaya a adquirir. Por ejemplo, en la medición de temperatura sólo se necesita tomar una muestra a la vez, en cambio en la medición de señales cardiacas es necesario obtener un número de muestras que permita describir la señal completa por lo menos durante un periodo.

3.1.1 Método Sencillo

El método sencillo, como se mencionaba anteriormente, consiste en adquirir muestra por muestra una señal analógica. Esto se puede hacer con un algoritmo como el que se muestra en la Figura 3.1. En este caso se debe configurar el dispositivo de adquisición y el canal respectivo, luego, la función “AI Sample Channel.vi” se encargará de adquirir la muestra y llevarla al indicador “Muestra”.

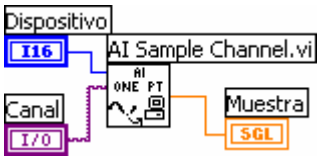


Figura 3.1. Adquisición sencilla.

El método de adquisición no *buffereado* tiene una variación que permite adquirir varias muestras y llevarlas a un indicador de tipo vector que las almacene. Esto se puede hacer añadiendo un bucle al algoritmo de la Figura 3.1 para repetir este proceso en un periodo de tiempo dado por una demora de software o por un reloj. Este reloj puede ser interno (lo

posee la TAD[12]) o externo (se debe conectar a la tarjeta, habiendo deshabilitado previamente el reloj interno). Una adquisición de este tipo con demora por software se muestra en la Figura 3.2.

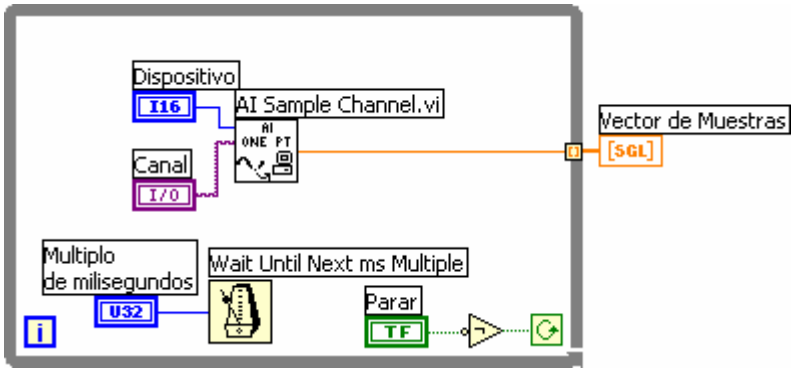


Figura 3.2. Variación del método sencillo.

El problema que aparece con las demoras por software es que no son muy precisas, lo que lleva a que la frecuencia de muestreo no se mantenga constante, en cambio con el reloj, el tiempo de muestreo es más preciso.

Sin embargo, al almacenar las muestras en un vector que ocupa una parte de la memoria principal, se está utilizando un método *buffereado*, con la diferencia de que el *buffer* no se establece antes de empezar la adquisición. Esta forma de adquirir presenta un problema con la adquisición continua de señales. Cada vez que se termina un ciclo de adquisición, se produce una demora que perjudica la señal, observándose un tiempo muerto como el de la Figura 3.3.

[12] TAD. Abreviatura de Tarjeta de Adquisición de Datos.

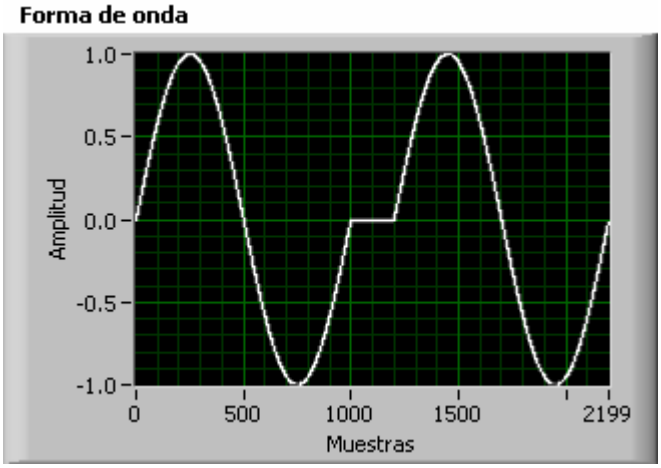


Figura 3.3 Ilustración de un tiempo muerto en la adquisición no *buffereada*.

3.1.2 Método *Buffereado*

El método *buffereado* consiste en adquirir un grupo de muestras y luego almacenarlas en un *buffer* localizado en memoria principal.

Existen dos métodos de adquisición *buffereada*: La simple y la circular.

El método *buffereado* simple localiza un *buffer* en la memoria RAM[13] del computador para almacenar las muestras que se van adquiriendo y sólo cuando se llena por completo o el número de muestras a adquirir se termina, se pueden procesar las muestras. Por otro lado, el *buffer* circular permite acceder a los datos mientras se adquieren. Un algoritmo como el que se muestra en la Figura 3.4 se puede utilizar como una adquisición *buffereada* simple y el de la Figura 3.5 como una adquisición *buffereada* circular.

[13] RAM. Abreviatura de *Random Access Memory.*

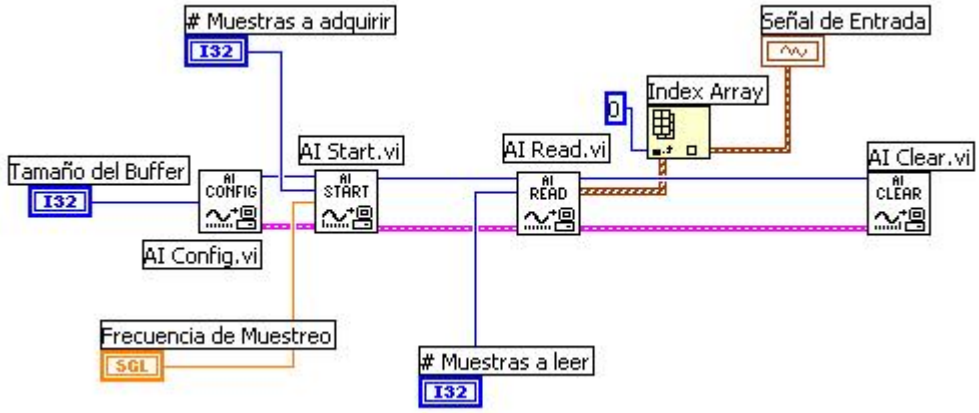


Figura 3.4. Adquisición *buffereada* simple.

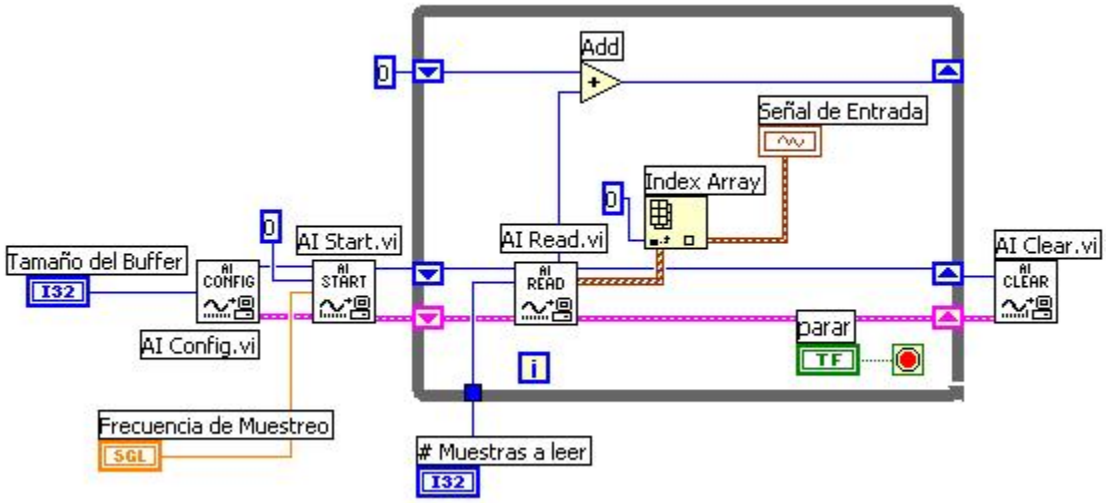


Figura 3.5. Adquisición *buffereada* circular.

El tipo de adquisición utilizado en este proyecto fue el *buffereado* circular, por eso a continuación, se explica detalladamente este método.

3.1.2.1 Adquisición *Buffereada* Circular

La adquisición *buffereada* circular, al igual que la simple guarda los datos que se van adquiriendo en un *buffer*, con la diferencia de que este es circular.

Un *buffer* circular es aquel que permite que una aplicación pueda acceder a los datos que contiene este mientras que él sigue recibiendo datos de otra aplicación. La Figura 3.6 muestra cómo trabaja un *buffer* circular.

En esta Figura se ve cómo los datos se van desplazando hacia la derecha. Los datos que entran primero son los que primero salen del *buffer*. Esta forma de guardar y recuperar los datos recibe el nombre de FIFO[14].

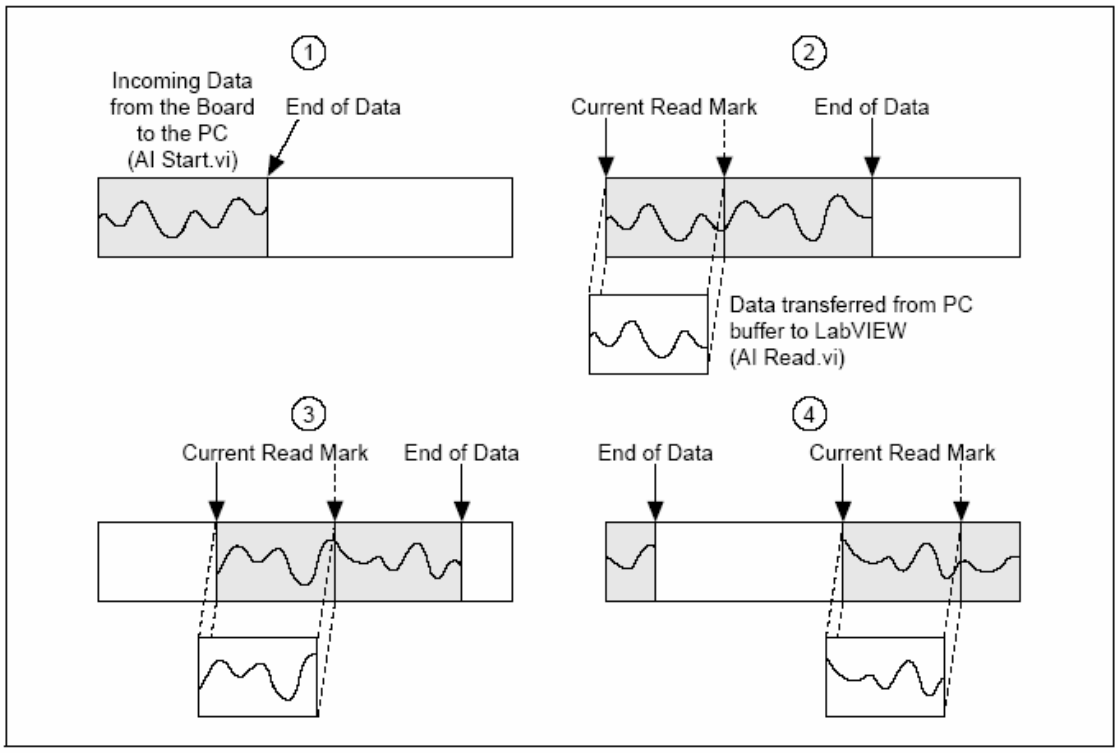


Figura 3.6. Cómo trabaja un *buffer* circular. Tomada de *LabView Measurements Manual*.

Al observar la Figura 3.6, el lector puede ver que el primer paso que sigue la adquisición *buffereada* circular es almacenar una cantidad de datos en el *buffer*. Esta operación la ejecuta la aplicación *AI Start.vi* de LabVIEW. El segundo paso consiste en transferir una cantidad de datos a una aplicación arbitraria de LabVIEW. Esta operación la produce la

[14] FIFO. Abreviatura de *First In First Out.*

rutina *AI Read.vi*. El tercer y el cuarto paso muestran cómo los datos se van desplazando, permitiendo que el *buffer* se llene y desocupe constantemente.

En la adquisición *buffereada* circular, comúnmente se presentan dos problemas. Uno de ellos ocurre cuando una aplicación recupera datos del *buffer* a una velocidad tan grande que lo desocupa por completo, esto provoca que la aplicación aguarde un tiempo no deseado para que los datos se establezcan en el *buffer* nuevamente. Este inconveniente se puede reparar aumentado el tamaño del *buffer* y/o la frecuencia de muestreo.

Otro problema se presenta cuando la aplicación recupera los datos del *buffer* a una velocidad tan pequeña, que hace que este se llene y los datos nuevos que entran a este se sobrescriban. Este inconveniente se soluciona disminuyendo la frecuencia de muestreo.

Un ejemplo común de la situación descrita anteriormente, se produce cuando el usuario interactúa con la aplicación que procesa los datos adquiridos. En este caso, la velocidad humana para resolver operaciones, como por ejemplo, introducir un nombre de un archivo, es muy lenta comparada con la velocidad de muestreo de una tarjeta de adquisición de datos. Esto ocasiona que el *buffer* se llene y los datos sean sobrescritos. Este problema se soluciona eliminando cualquier operación que necesite la interacción con el usuario mientras se ejecuta la adquisición, ya que, disminuir la frecuencia de muestreo, no bastaría para corregir esta falla.

Después de haber visto con detalle los diferentes tipos de adquisición se puede concluir que para realizar la adquisición de la señal de forma continua se debe utilizar la técnica *buffereada* circular. Este tipo de adquisición utiliza un grupo de funciones que se explican a continuación.

3.1.3 Funciones que se Utilizan en la Adquisición *Buffereada*

El proceso de adquisición de señales mediante Labview es una tarea aparentemente sencilla, ya que este lenguaje contiene librerías dedicadas a la adquisición de señales

analógicas. De esta forma, el programador sólo debe utilizarlas, según la necesidad que tenga.

La adquisición *buffereada* circular que se utilizó en este proyecto, se puede implementar mediante cuatro funciones de adquisición integradas en Labview. Estos son *AI Config.vi*, *AI Start.vi, AI Read.vi y AI Clear.vi.* A continuación se explican detalladamente las aplicaciones de adquisición mencionadas.

3.1.3.1 AI Config.vi

Esta función se utiliza para configurar la adquisición de señales analógicas. Entre los parámetros que se pueden configurar en una adquisición a través de *AI Config.vi* están: el dispositivo de adquisición, el canal de adquisición, el rango límite de la señal de entrada y el tamaño del *buffer*, como los más importantes.

En la Figura 3.7 se muestra esta rutina con sus respectivas entradas y salidas.

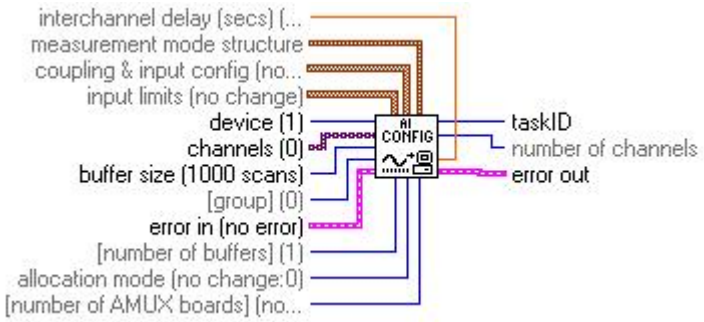


Figura 3.7. *AI Config.vi.* Tomada de *LabView Help*

A través de la salida *taskID, AI Config.vi* envía un número que identifica la configuración del dispositivo de adquisición de datos. Este valor les permite a las demás funciones, identificar los valores de los parámetros de adquisición que se van a utilizar.

El diseñador puede dar nombre al dispositivo de adquisición de datos y sus canales respectivos, a través de la aplicación *Measurements and automation*, incluida en el

paquete *National Instruments LabVIEW 6i*. Allí también se pueden configurar otros parámetros como el rango de trabajo de la TAD, el modo de conexión de las señales de entrada, entre otros.

A continuación se explican las entradas y salidas más importantes de esta función.

- ***Device*:** es el número que se asigna al dispositivo de adquisición de datos durante la configuración.

- ***Channels*:** especifica el conjunto de canales de entrada análoga. El orden en el que se definen los canales, es el mismo en cual son leídos durante la adquisición. Los canales se deben organizar en un arreglo de *strings*. Se puede dar un nombre a cada canal o dar un nombre a un conjunto de canales, separándolos por comas. También, se puede hacer una combinación de los dos métodos. Si el usuario no desea adjudicar un nombre al canal o canales que va a utilizar, puede simplemente numerarlos empezando desde el canal 0 hasta el número máximo de canales que tenga la tarjeta. La tarjeta *PCI 1200* posee 8 canales de entrada, y se numeran del 0 al 7.

- ***Buffer size*:** es el número de muestras que puede almacenar el *buffer*.

- ***Error in*:** describe las condiciones de error ocurridas antes de la ejecución de esta rutina. Si un error ha ocurrido anteriormente, esta aplicación retorna un valor de error a través de la salida *error out.*

- ***TaskID*:** genera un número que identifica la configuración del dispositivo de adquisición.

- ***Error out:*** contiene la información de error. Si la entrada *error in* contiene un error, la salida *error out* describe ese error. De otra forma, si en el instrumento virtual actual se presenta un error, *error out* describe el estado actual del error. Si no hay errores, *error out* genera un valor indicando que no hay error y la rutina puede seguirse ejecutando sin problema.

3.1.3.2 AI Start.vi

Esta función comienza una adquisición *buffereada* y configura la frecuencia de muestreo (*scan rate*) y el número de muestras a adquirir. En la Figura 3.8 se observa la función *AI Start.vi* con sus respectivas entradas y salidas.

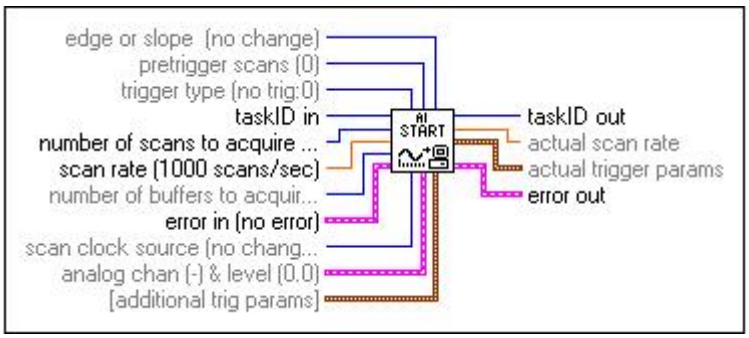


Figura 3.8. *AI Start.vi.* Tomada de *LabView Help*.

A continuación se presentan las entradas y salidas de esta rutina con sus respectivas definiciones.

- ***TaskID in**:* identifica el grupo de canales y el tipo de operación de entrada de datos.

- ***Number of scans to acquire:*** es el número de muestras a adquirir. Un *scan* es una muestra y esta a su vez, es un punto por canal. Cuando la entrada es uno, LabVIEW adquiere exactamente un *buffer* de datos. El tamaño del *buffer* lo determina la entrada *Buffer size* de *AI Config.vi.* Cuando la entrada es cero, LabVIEW adquiere datos indefinidamente hasta que se termine la adquisición por medio de *AI Clear.vi.*

- ***Scan rate:*** es la velocidad a la cual adquiere datos el dispositivo. Está dada en [muestras por segundo] y es equivalente a la frecuencia de muestreo por canal. El

valor predeterminado de esta entrada es 1000 muestras/s. Una entrada de -1 produce una frecuencia de reloj aleatoria. Se puede usar este valor cuando la frecuencia de reloj del dispositivo no sea programable, como por ejemplo en las series 43XX y 40XX de las TAD *National Instruments*. Si el valor ingresado es cero, LabVIEW deshabilita el reloj interno de la tarjeta y permite usar un reloj externo. Para mayor información acerca de las fuentes de reloj, se debe consultar la ayuda de *AI Clock Config.vi*.

- ***TaskID out:*** Tiene el mismo valor de *TaskID in*.

La entrada *error in* y la salida *error out* cumplen las mismas funciones que en *AI Config.vi*.

3.1.3.3 AI Read.vi

Esta aplicación lee un número específico de datos del *buffer* de adquisición. La principal salida de esta función es el vector o forma de onda que contiene los datos recuperados. La rutina *AI Read.vi* se muestra en la Figura 3.9.

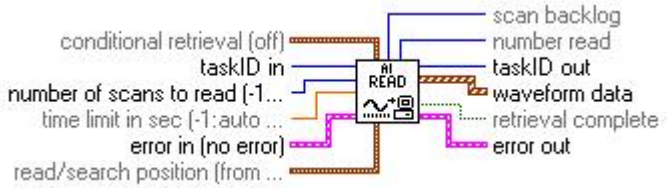


Figura 3.9. *AI Read.vi.* Tomada de *LabView Help*.

A continuación se presentan las entradas y salidas de esta función con sus respectivas definiciones.

- ***Number of scans to read:*** es el número de muestras que la aplicación recupera del *buffer* de adquisición. La entrada predeterminada es -1, la cual hace que el número de muestras a leer se vuelva igual al número de muestras a adquirir de *AI*

Start.vi. Si el número de muestras a leer es -1 y el número de muestras a adquirir es 0; LabVIEW escribe el número “100” en la entrada *number of scans to read.*

- ***Waveform data:*** Es un arreglo de datos de una dimensión, que contiene las formas de onda provenientes de los canales de entradas analógicas, con sus unidades previamente configuradas en *measurements & automation*.

3.1.3.4 AI Clear.vi

Esta aplicación termina la adquisición de señales. La Figura 3.10 muestra la función *AI Clear.vi* con sus respectivas entradas y salidas.

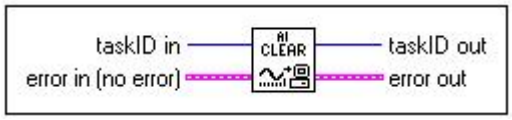


Figura 3.10. *AI Clear.vi.* Tomada de *LabView Help*.

Cuando el bloque *AI Clear.vi* es ejecutado, el valor que entra por *taskID in* se convierte en cero y sale por *taskID out* eliminando la tarea de adquisición.

Habiendo explicado todo lo relacionado con la adquisición de la señal, se procede a explicar las funciones que se utilizaron en los algoritmos de filtrado.

3.2 FUNCIONES DE DISEÑO DE FILTROS

Las funciones que se utilizan para diseñar los filtros son tres: *Digital IIR Filter.vi, Digital FIR Filter.vi, e IIR Filter.vi* y se utilizan cada una para diseñar filtros IIR o FIR según corresponda.

3.2.1 Digital IIR Filter.vi

Como su nombre lo dice, esta función permite filtrar una o varias señales con un filtro IIR. Este instrumento virtual permite diseñar un filtro para cada señal de entrada o uno para todas. Cada señal se debe introducir por un canal de entrada de la TAD. En el caso de la TAD PCI 1200 que posee 8 canales, se pueden introducir 8 señales referenciadas al mismo punto. Si la referencia de cada señal es diferente, los canales se deben tomar en forma diferencial, por lo que se obtendrían solamente 4 canales. En el manual de la TAD se explica el modo de conexión para cada caso.

Las señales de entrada se deben llevar a un arreglo de formas de onda para poder introducirlas al filtro. Así mismo si se va a utilizar un filtro para cada canal, se deben introducir las especificaciones en un arreglo. El diagrama de esta función se puede ver en la Figura 3.11.

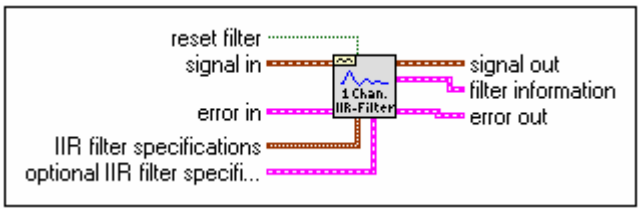


Figura 3.11. *Digital IIR Filter.vi* Tomada de *LabView Help*.

Las diferentes topologías que puede manejar esta función son: Butterworth, Chevyshev, Elíptica y Bessel y permite diseñar los cuatro tipos de filtros: Pasa altas, pasa bajas, pasa banda y para banda.

Con sólo dar las especificaciones del filtro, esta función diseña el filtro, genera la respuesta en frecuencia de magnitud y fase. Si una señal se conecta a la entrada *signal in*, por la salida *signal out* se generará la señal filtrada.

Las entradas y salidas más importantes de esta función se explican a continuación

- ***Signal in:*** Es la señal de entrada

- ***IIR filter specifications:*** Contiene los parámetros de diseño del filtro y se muestran en la Figura 3.12.

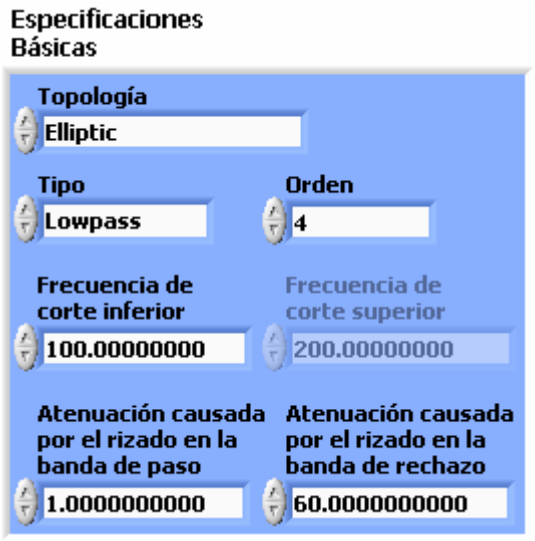


Figura 3.12. Parámetros de diseño de un filtro IIR. Tomada de LabVIEW

- ***Optional IIR filter specification:*** Contiene la información necesaria para computar el orden del filtro IIR y se pueden ver en la Figura 3.13.

Figura 3.13. Parámetros para computar el orden de un filtro IIR. Tomada de LabVIEW

- ***Signal out:***: Es la señal filtrada

- ***Filter information:*** Contiene la magnitud, fase y orden del filtro.

3.2.2 Digital FIR Filter.vi

Esta función es muy similar a la anterior con la única diferencia de que las especificaciones se deben dar para un filtro FIR. Los parámetros de diseño de un filtro FIR se muestran en la Figura 3.14.

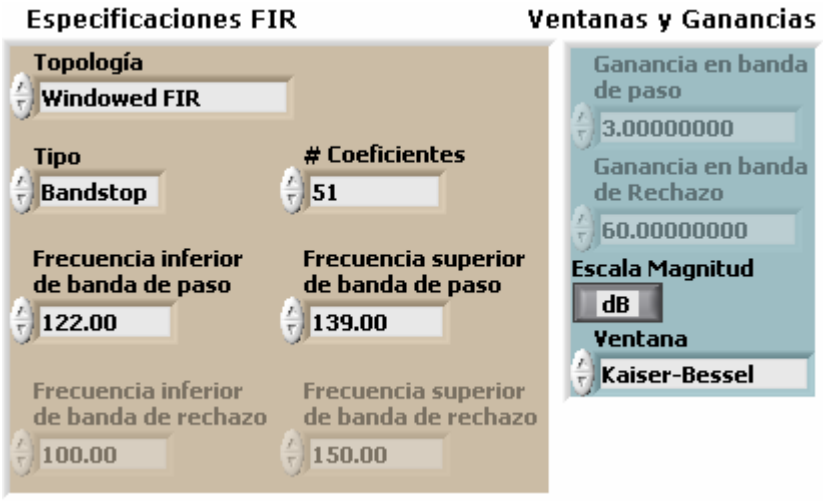


Figura 3.14. Parámetros de diseño de un filtro FIR. Tomada de LabVIEW

Esta función permite diseñar dos topologías de filtros FIR: rizado constante y ventaneado. Además posee una opción llamada *FIR by specification* que computa el orden del filtro para unas especificaciones dadas.

3.2.3 IIR Filter.vi

Esta función filtra una secuencia de entrada usando la forma directa dada por los coeficientes de la ecuación de diferencias de un filtro. El diagrama se muestra en la Figura 3.15.

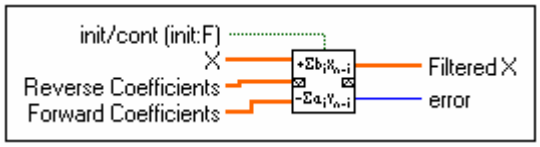


Figura 3.15. *IIR Filter.vi.* Tomada de LabView *Help*.

Las entradas y salidas más importantes de esta función se explican a continuación.

- ***X:*** Es la secuencia de entrada. También acepta una forma de onda.

- ***Reverse Coefficients:*** Son los coeficientes que multiplican la secuencia de salida.

- ***Forward Coefficients:*** Son los coeficientes que multiplican la secuencia de entrada.

- ***Filtered X:*** Es la secuencia filtrada.

Esta función también se puede usar para filtros FIR introduciendo sólo los coeficientes que multiplican la secuencia de entrada.

Después de haber conocido las características de las funciones de filtrado, se procede a resolver por medio de estas funciones algunos ejemplos que se solucionaron en el capítulo 1 con algoritmos básicos.

3.3 EJEMPLOS DE DISEÑO

Ejemplo 3.1 Diseño de un filtro FIR pasa bajas con ventana rectangular

En el ejemplo 1.1 se pidió diseñar un filtro FIR con ventana rectangular, pasa bajas, frecuencia de corte de 100Hz, orden 2 y frecuencia de muestreo de 1kHz.

Los parámetros de diseño que se deben introducir en la función *Digital FIR Filter.vi* de LabVIEW son los que se muestran en la Figura 3.16.

La magnitud y fase resultante se muestran en la Figura 3.17

Si estas gráficas se comparan con las mostradas en la solución del ejemplo 1.1 se podrá observar que los resultados son los mismos.

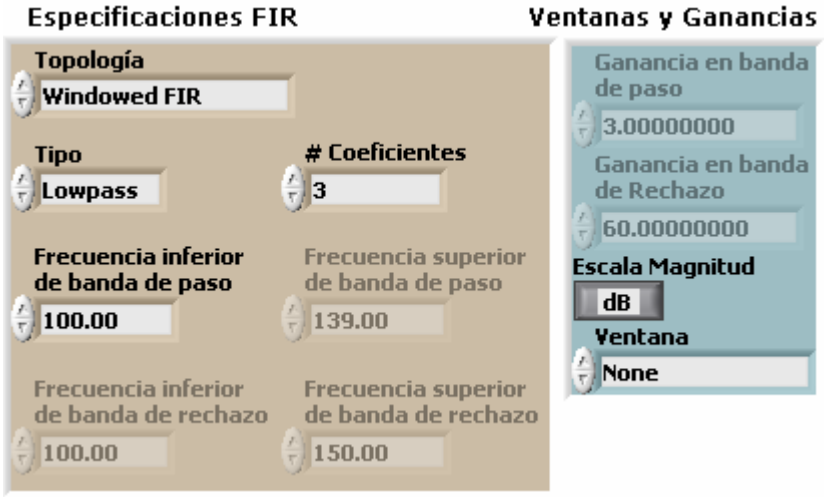


Figura 3.16. Parámetros de diseño de un filtro FIR de orden 2, pasa bajas, frecuencia de corte de 100Hz y frecuencia de muestreo de 1kHz. Tomada de LabVIEW.

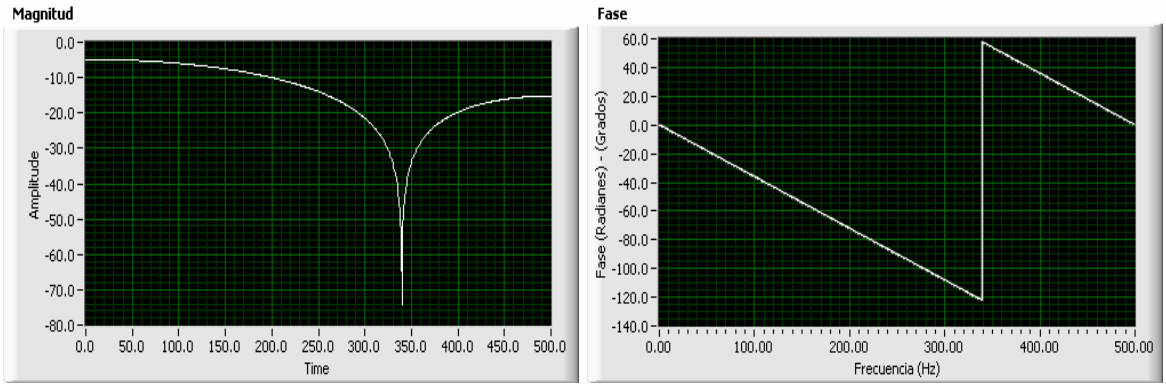


Figura 3.17. Magnitud y fase del filtro FIR de 2º orden del ejemplo 3.1.

Ejemplo 3.2 Diseño de un filtro FIR pasa bajas con ventana de Kaiser

En el ejemplo 1.2 se diseñó un filtro paso bajo con ventana de Káiser, frecuencia de paso de 6kHz y frecuencia de rechazo de 9kHz, rizado en la banda de paso de 0.1 y de 0.01 en la banda supresora. La frecuencia de muestreo fue de 30kHz.

Basados en los cálculos hechos en el ejemplo 1.2 se tiene que la frecuencia de corte del filtro es de 7.5kHz y el orden es de 22.

Los parámetros de diseño de este filtro se muestran en la Figura 3.18.

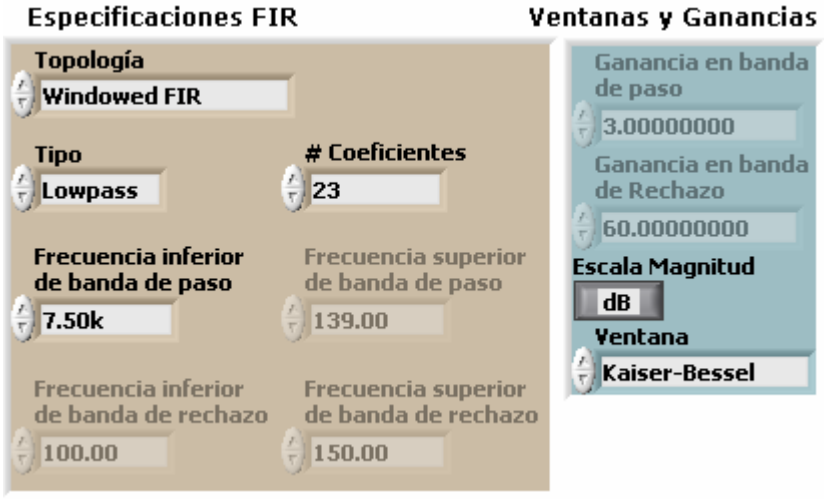


Figura 3.18. Parámetros de diseño de un filtro FIR pasa bajas de orden 22 utilizando la ventana de Kaiser.

Las respuestas de magnitud y fase del filtro diseñado se muestran en la Figura 3.19. Al comparar estos resultados con los obtenidos en el ejemplo 1.2, se encuentra que son muy similares.

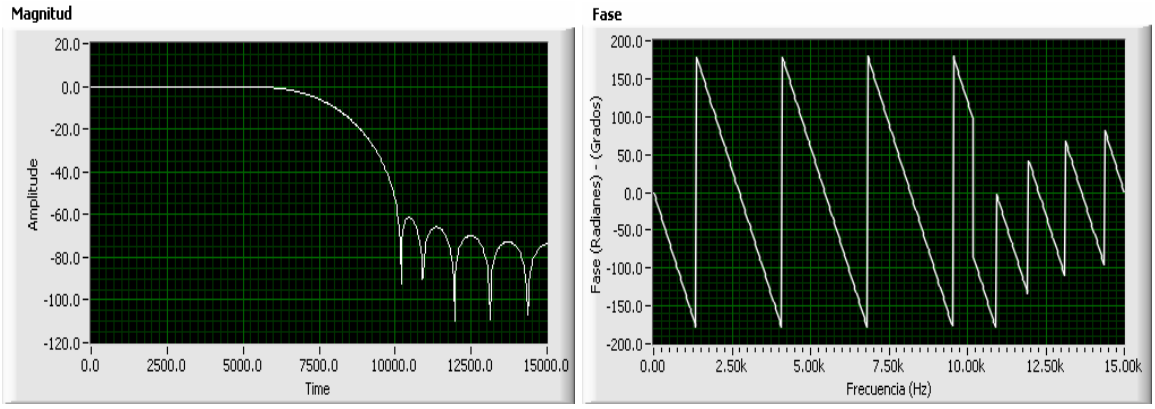


Figura 3.19. Magnitud y fase del filtro FIR de orden 22 del ejemplo 3.2.

Ejemplo 3.3 Diseño de un filtro IIR pasa bajas de topología Butterworth

En el ejemplo 1.4 se pidió diseñar un filtro digital paso bajo IIR con una frecuencia de muestreo de 10kHz, frecuencia de corte de 1034Hz y al menos una atenuación de 10dB a la frecuencia de 2312Hz.

Con las especificaciones opcionales se puede computar el orden del filtro. Estas se pueden ver en la Figura 3.20.

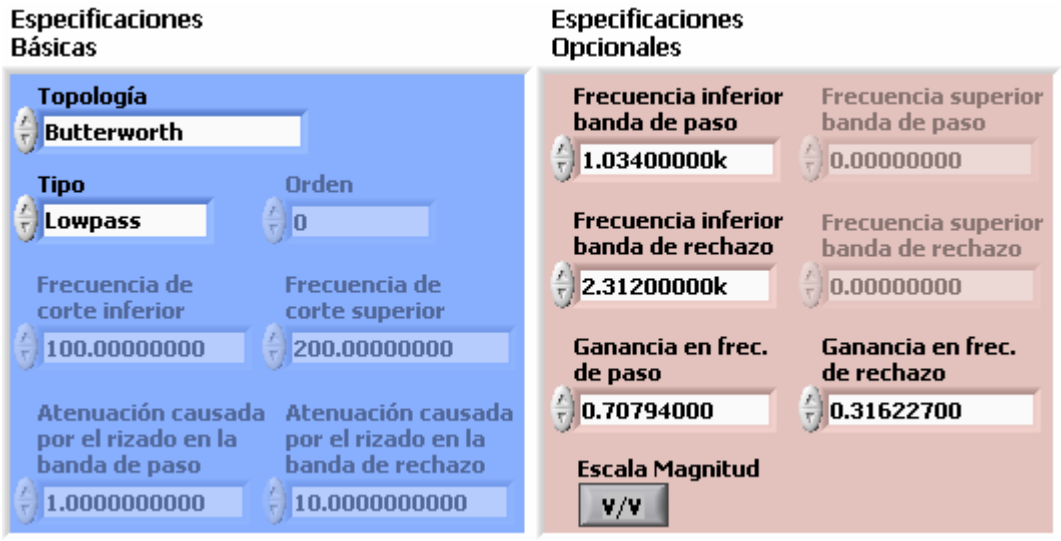


Figura 3.20. Parámetros de diseño del filtro IIR del ejemplo 3.3.

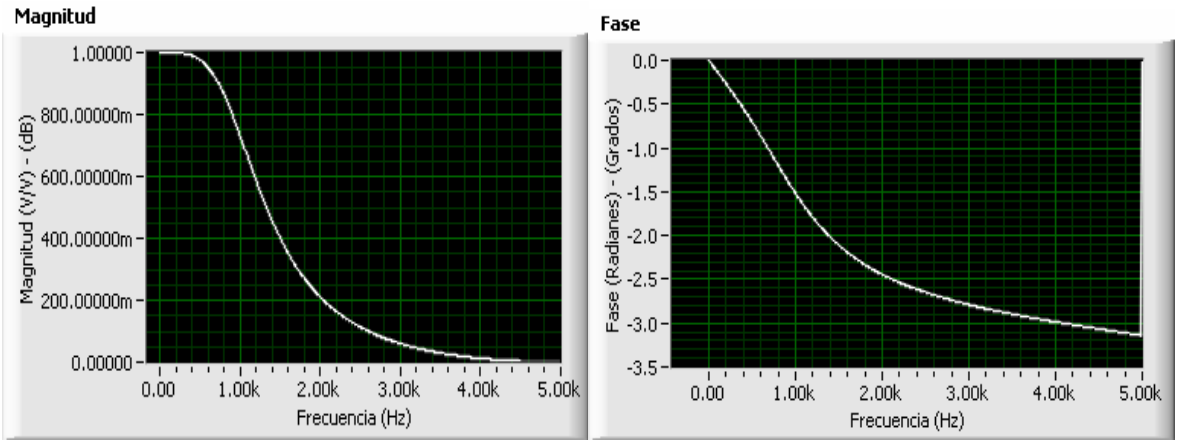


Figura 3.21. Magnitud y fase del filtro IIR de orden 2 del ejemplo 3.3.

Las respuestas de magnitud y fase se muestran en la Figura 3.21. Al comparar estos resultados con los del ejemplo 1.4 se puede ver que son prácticamente los mismos.

En este proyecto se desarrolló una aplicación que compacta las funciones de filtrado y adquisición de señales con el fin de realizar un filtrado digital en tiempo real cubriendo el proceso completo que se implementó en tres etapas: diseño, adquisición y procesamiento, tal y como se muestra en la Figura 3.22.

Figura 3.22. Etapas del filtrado digital en tiempo real.

La guía de usuario de esta aplicación se muestra en el anexo B.2. y el diagrama de bloques no se muestra ya que al ser una recopilación de lo visto en este capitulo, generaría redundancia, sin embargo los instrumentos virtuales que componen esta aplicación se encuentran incluidos en el CD que acompaña a este texto. En el anexo B.2 se mencionan los archivos que deben estar presentes a la hora de ejecutar la aplicación y el espacio que ocupan en disco duro.

Además de la aplicación en tiempo real, se diseñó una en modo simulación que le permite al usuario generar diferentes tipos de señales, diseñar filtros y probarlos con estas señales, todo únicamente utilizando LabVIEW.

Ante la ausencia de una TAD, la aplicación **tiempo real.vi** permite diseñar filtros pero es imposible que estos sean probados con señales. Con **simulación.vi**, esto se puede hacer desde cualquier PC que tenga instalado el *software* LabVIEW 6.I y lograr prever lo que pasaría en tiempo real.

Esta aplicación utiliza los mismos *subvis* de diseño y procesamiento que tiempo real.vi, por lo que en apariencia son prácticamente iguales.

4. PRUEBAS REALIZADAS Y RESULTADOS OBTENIDOS

En este capítulo se muestran los resultados obtenidos de las pruebas realizadas con las aplicaciones desarrolladas en este proyecto, estas son el generador de señales arbitrarias y el filtrado digital en tiempo real. Las pruebas se realizaron en un PC hp-compaq con procesador AMD Athlon XP 2000+ (frecuencia de reloj 1.66GHz) y 256MB de memoria RAM. Las pruebas se dividen en tres partes. En la primera se generan diferentes formas de ondas utilizando el generador de señales arbitrarias, en la segunda se diseñan filtros de diferentes topologías y en la tercera parte se evalúan la adquisición y filtrado digital en tiempo real con formas de onda y espectros conocidos. El objetivo de este capítulo, es por tanto, mostrar la utilidad de las dos aplicaciones diseñadas, tanto en conjunto como individualmente.

4.1 PRUEBAS AL GENERADOR DE SEÑALES ARBITRARIAS

Para realizar las pruebas al generador se midieron las formas de onda con dos sistemas diferentes el osciloscopio *FLUKE* 105 Scopemeter y con el DSP TMS320CV5402 Starter Kit de Texas Instruments, con el objetivo de tener resultados más confiables.

4.1.1 Pruebas con el Fluke Scopemeter

La primera prueba se realizó utilizando un osciloscopio *FLUKE 105 Scopemeter* conectado al canal 0 de la tarjeta de adquisición PCI 1200 (entre los pines 10 y 11).

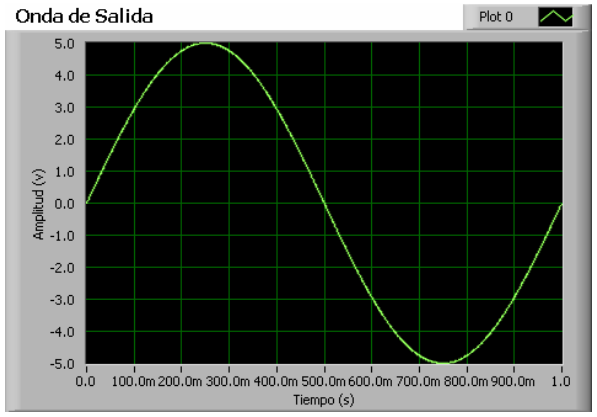
Con el osciloscopio *FLUKE* se realizó una prueba a cada opción principal del generador. Estas son: generar una señal estándar de librería, generar una señal de una fórmula

matemática, generar una señal dibujada con el mouse, generar una señal cargada de un archivo de texto, generar una adición y una diferencia entre dos señales. También se probó la generación simultánea por el canal 0 y el canal 1. Todas las pruebas se realizaron configurando el generador con una rata de generación de 1000 muestras/segundo y un tamaño del *buffer* de 1000 muestras. Se establece también modo de generación continuo para obtener señales periódicas.

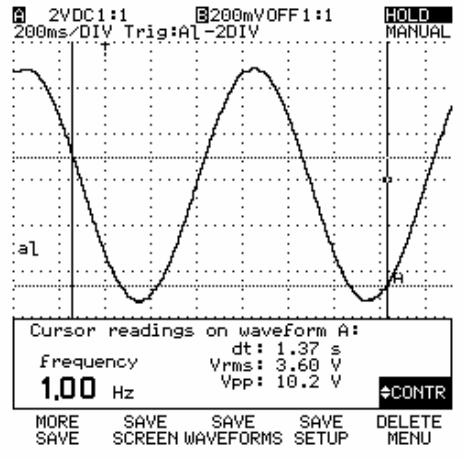
- **Función librería**

Siguiendo los pasos explicados en el anexo B, se editó una señal senoidal de 1Hz con amplitud de 5V sin *offset*. En la Figura 4.1a se observan los valores de los controles en el panel frontal del subprograma *Librería* para editar esta señal. En la Figura 4.1b se observa la previsualización de la onda de salida para la señal editada. Se observa que el arreglo de datos corresponde a un ciclo de una senoidal y la información de tiempo de la gráfica indica que es una señal de 1Hz y 5V de amplitud, sin *offset*. En la Figura 4.1c se observa que la señal medida en el FLUKE es la esperada, una senoidal de 1Hz con 10 Vpp.

(a)



(b)

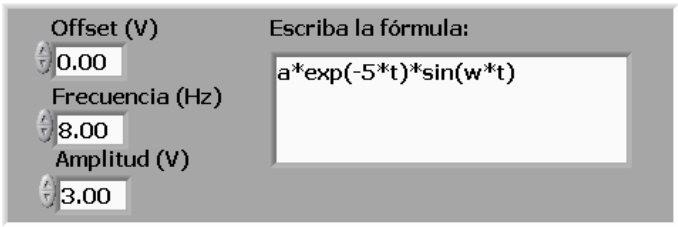


(c)

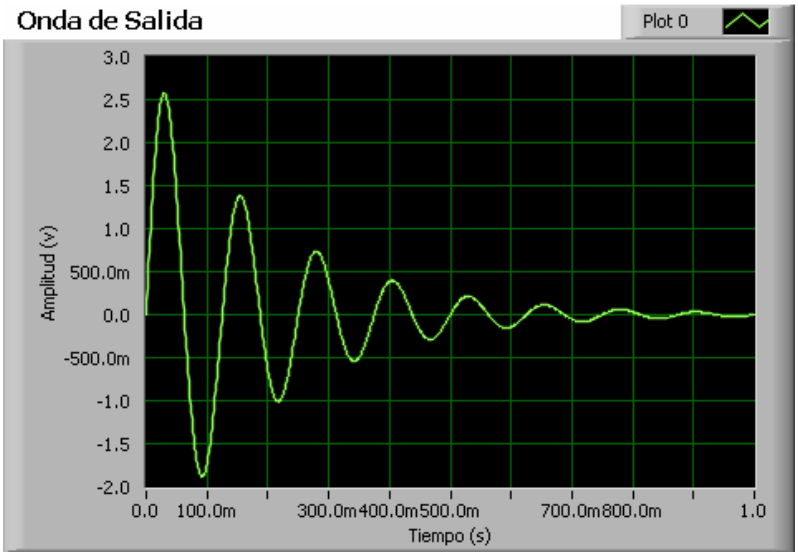
Figura 4.1 Resultados de la prueba a la opción Función de Librería. **a)** Valores de los controles **b)** Previsualización de la señal **c)** Señal medida en el *FLUKE.*

- **Función de Fórmula**

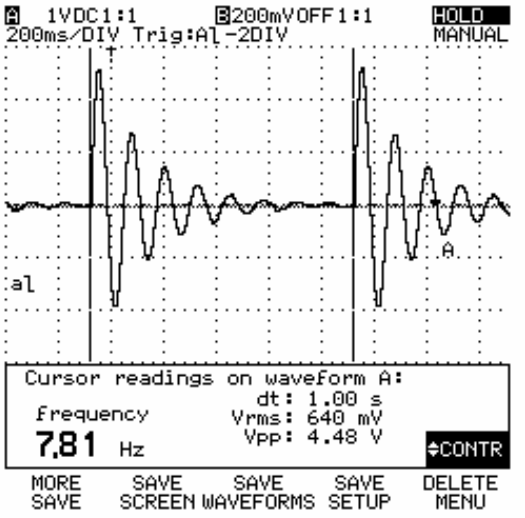
Siguiendo los pasos explicados en el anexo B se editó la señal a*exp(-5*t)*sin(w*t). En la Figura 4.2a se observan los valores de los controles para editar esta señal en el panel frontal del subprograma *Editar Form.* Reemplazando estos valores se tendría la fórmula $3*e^{-5t}$ *sen(2*pi*8*t). Esta función no es periódica, pero cabe recordar que en modo continuo se repetirá la forma de onda observada en la Figura 4.2b y por tanto esta onda previsualizada representará un ciclo de una señal periódica. En la Figura 4.2c se puede comprende mejor esto ya que en el *FLUKE* se observó una señal periódica, donde un período es el representado por la Figura 4.2b. Las lecturas entregadas por el *FLUKE* están de acuerdo a lo esperado. Cada pulso de senoidal amortiguada tiene una duración de 1s (dt=1s) y la señal tiene un voltaje pico a pico de 4.48V. Teóricamente esta señal es de 1Hz de frecuencia, sin embargo el FLUKE no logra captar esta frecuencia teórica debido a las oscilaciones de 8Hz de la señal senoidal amortiguada y por tanto despliega una lectura de frecuencia de 7.81Hz.



(a)



(b)



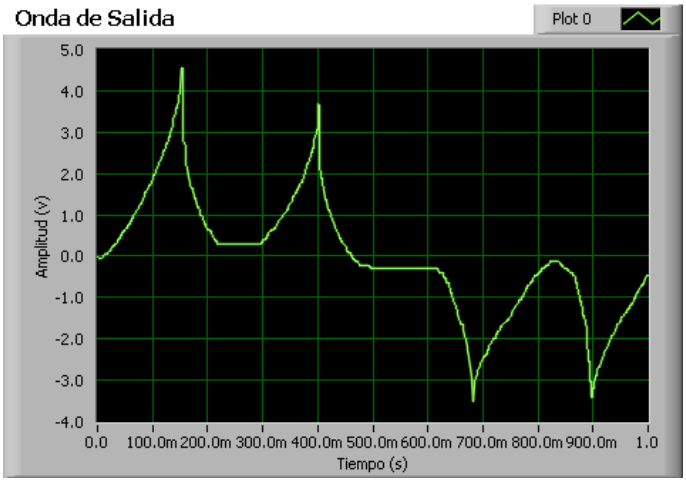
(c)

Figura 4.2 Resultados de la prueba con la función de fórmula. **a)** Valores de los controles **b)** Previsualización de la señal **c)** Señal medida en el *FLUKE*.

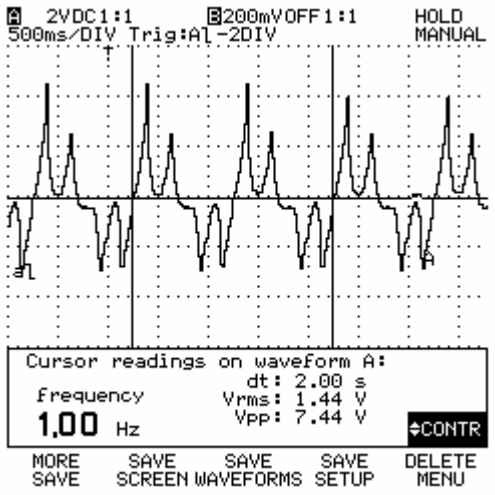
- **Función de Mouse**

Siguiendo los pasos explicados en el anexo B se editó la forma de onda dibujada con el mouse que se observa en la Figura 4.3a.

Al iniciar la generación se observó en el *FLUKE* la señal de la Figura 4.3b. La lectura del *FLUKE* está de acuerdo con la visualizada en la Figura 4.3a. Esta es una señal periódica bipolar de 1Hz con un voltaje pico a pico de 7.44V.



(a)



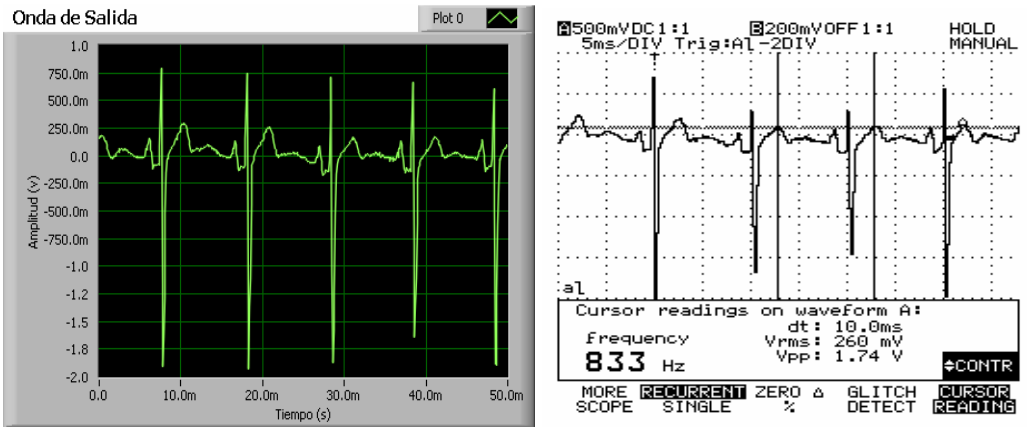
(b)

Figura 4.3. Resultados de las pruebas a la opción Función de Mouse. **a)** Previsualización de la señal **b)** Señal medida en el *FLUKE*.

- **Señal Cargada de Archivo de Texto.**

Esta prueba se realizó cargando 500 muestras del archivo de texto ECG.txt. Este archivo de texto es una modificación a una simulación del ritmo cardíaco realizada por el codirector del proyecto Ing. Javier Gonzáles. Para esta prueba se modificó la rata de generación a 10000 muestras/s para mejorar la resolución horizontal de los pulsos cardíacos. La forma de onda previsualizada se observa en al Figura 4.4a.

En la Figura 4.4b se observa la señal captada en el *FLUKE*. Las lecturas son correctas ya que la duración de un pulso cardíaco es de aproximadamente 10ms, tal y como se observa en la Figura 4.4a. El voltaje pico a pico entregado es de 1.74V, el cual no es estable en el *FLUKE* debido a los cambios abruptos en la amplitud de la señal generada.

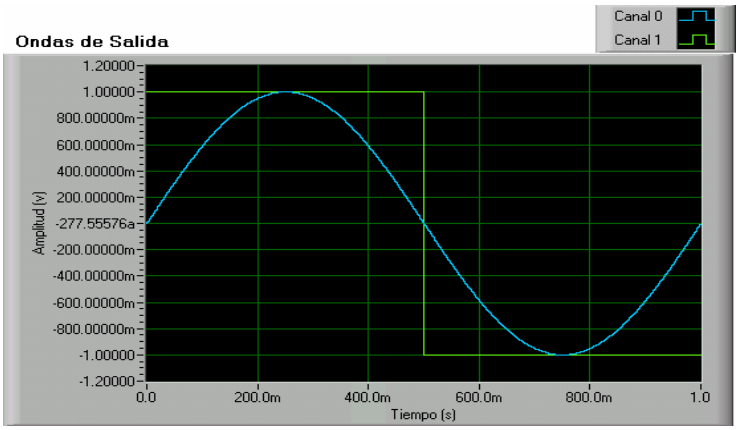


(a) **(b)**

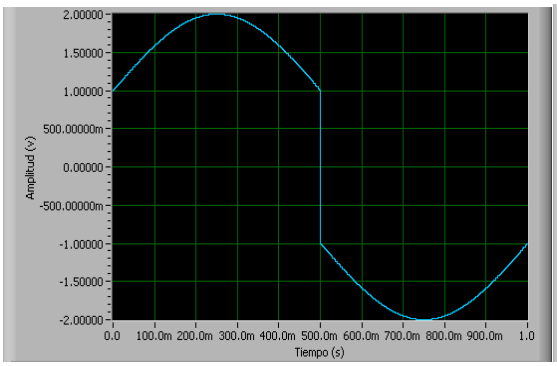
Figura 4.4. Resultados de las pruebas a la opción Cargar Señal. **a)** Previsualización de la señal de salida. **b)** Señal medida en el *FLUKE.*

- **Adición y Diferencia de Señales**

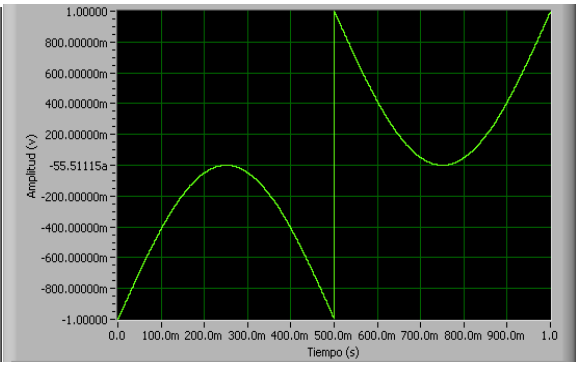
Para probar estas opciones se editaron las formas de onda que se observan en la Figura 4.5.


Ondas de Salida
Canal 0
Canal 1
1.20000
1.00000
800.00000m
600.00000m
400.00000m
200.00000m
-277.55576a
-200.00000m
-400.00000m
-600.00000m
-800.00000m
-1.00000
-1.20000
Amplitud (v)
0.0
200.0m
400.0m
600.0m
800.0m
1.0
Tiempo (s)

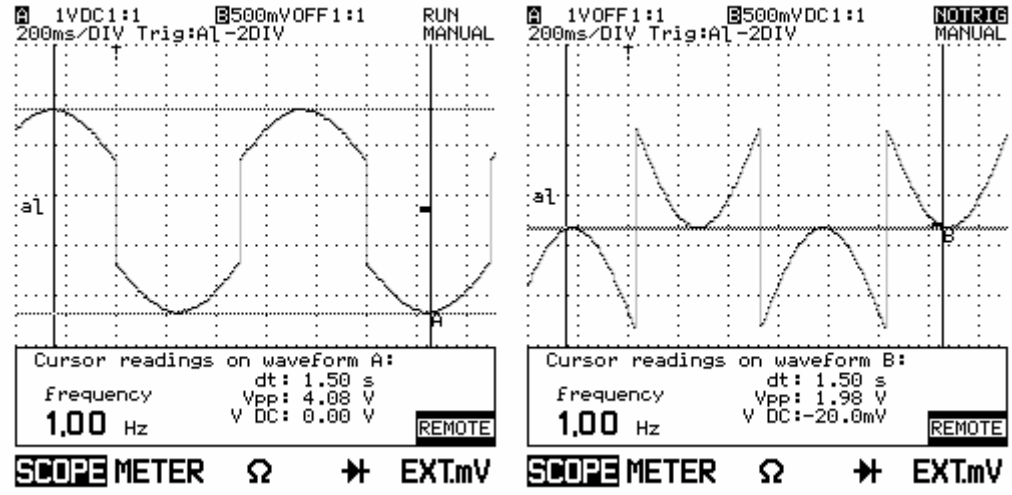
(a)


2.00000
1.50000
1.00000
500.00000m
0.00000
-500.00000m
-1.00000
-1.50000
-2.00000
Amplitud (v)
0.0
100.0m
200.0m
300.0m
400.0m
500.0m
600.0m
700.0m
800.0m
900.0m
1.0
Tiempo (s)

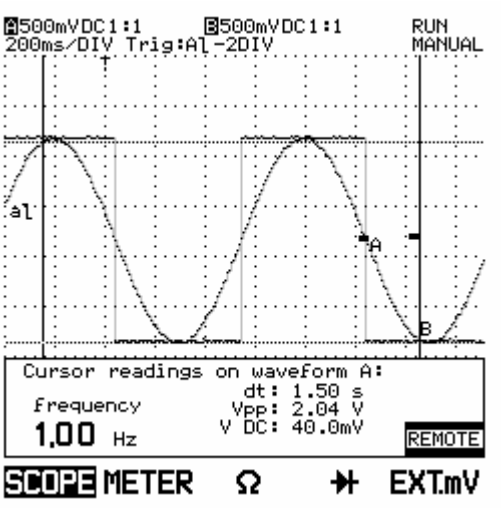
(**b**)


1.00000
800.00000m
600.00000m
400.00000m
200.00000m
-55.51115a
-200.00000m
-400.00000m
-600.00000m
-800.00000m
-1.00000
Amplitud (v)
0.0
100.0m
200.0m
300.0m
400.0m
500.0m
600.0m
700.0m
800.0m
900.0m
1.0
Tiempo (s)

(c)


A 1VDC1:1
B500mVOFF1:1
RUN
200ms/DIV Trig:Al-2DIV
MANUAL
Al
A
Cursor readings on waveform A:
dt: 1.50 s
Frequency
Vpp: 4.08 V
V DC: 0.00 V
1.00 Hz
REMOTE
SCOPE METER Ω EXT.mV
A 1VOFF1:1
B500mVDC1:1
NOTRIG
200ms/DIV Trig:Al-2DIV
MANUAL
Al
B
Cursor readings on waveform B:
dt: 1.50 s
Frequency
Vpp: 1.98 V
V DC:-20.0mV
1.00 Hz
REMOTE
SCOPE METER Ω EXT.mV

(d)



(e)

Figura 4.5 Resultados de la prueba de adición y diferencia de señales. **a)** Señales individuales **b)** Adición en el canal 0 **c)** Diferencia en el canal 1 **d)** Señales medidas en el *FLUKE.* **e)** Generación simultánea por el canal 0 y 1.

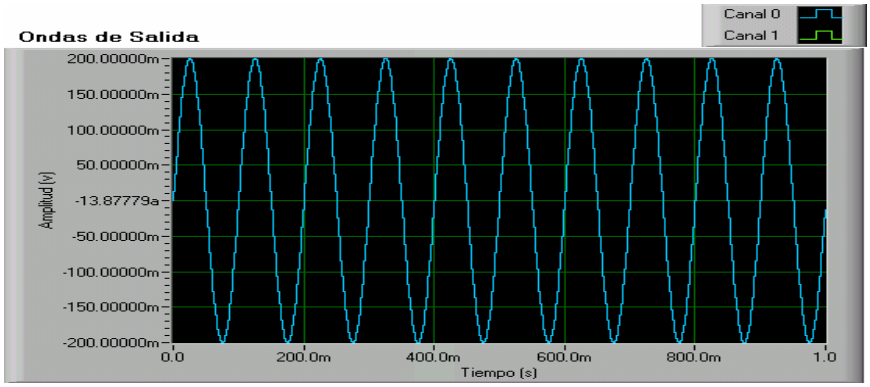
La Figura 4.5a representa la señal editada en el canal 0 (color azul) y la señal editada en el canal 1 (color verde). Las señales de las Figura 4.5b y 4.5c corresponden a la previsualización de las ondas de salida luego de escoger la opción de adición y diferencia respectivamente de las señales en la Figura 4.5a. En la Figura 4.5d se pueden observar las señales medidas por el *FLUKE.* Como se observa en la figura, la adición es una señal periódica con voltaje pico a pico de 4V, una frecuencia de 1Hz y sin *offset*. La diferencia es una señal periódica con voltaje pico a pico de 2V, una frecuencia de 1Hz y sin *offset*; por tanto los valores medidos corresponden a la señales editas. También se observa en la Figura 4.5e la generación simultánea de las señales observadas en la Figura 4.5b y 4.5c. La señal senoidal del canal 0 se midió por el canal A del *FLUKE* entre los pines 10 y 11; y la señal cuadrada se midió por el canal B del *FLUKE* entre los pines 10 y 12 de la bornera de la TAD PCI 1200. Se observa que las dos señales corresponden efectivamente a las señales editadas en la Figura 4.5a.

Con estos resultados terminan las pruebas a las opciones del generador de señales arbitrarias utilizando el oscilospio *FLUKE 105*. A continuación se realizaran otras

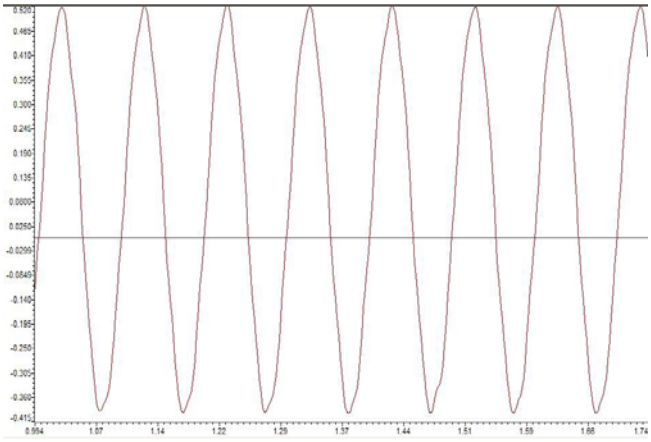
pruebas al generador de señales arbitrarias para confirmar los excelentes resultados obtenidos hasta ahora.

4.1.2 Pruebas DSP TMS320CV5402 Starter Kit de Texas Instruments

Estas pruebas se apoyaron en el proyecto de maestría “Procesamiento Digital de Señal Electrocardiográfica con Tecnología DSP Orientado al Análisis de Variabilidad de la Frecuencia Cardiaca” del Ing. Javier González Barajas. En la Figura 4.6a se observa una senoidal de 10Hz y 200mV de amplitud generada con la aplicación de generación de señales arbitrarias. En la Figura 4.6b se observa la señal adquirida con el sistema de adquisición basado en DSP con una frecuencia de muestreo de 2kHz y en la Figura 4.6c se observa su espectro en frecuencia.



(a)



(b)

(c)

Figura 4.6 Adquisición de una señal de 10Hz y 200mV. **a)** Señal generada **b)** Señal adquirida con el DSP **b)** Espectro en frecuencia procesado por el DSP

Otra prueba fue la adquisición de la suma de tonos y(t)=0.5*sin(6.2831*30*t)+0.5*sin(6.2831*50*t)+0.5*sin(6.2831*58*t)+0.5*sin(6.2831*60*t)+ 0.5*sin(6.2831*62*t)+0.5*sin(6.2831*70*t) [V] de la cual se puede comparar en la Figura 4.7 el espectro de potencia procesado por el sistema basado en DSP y por el sistema de adquisición desarrollado en este proyecto.

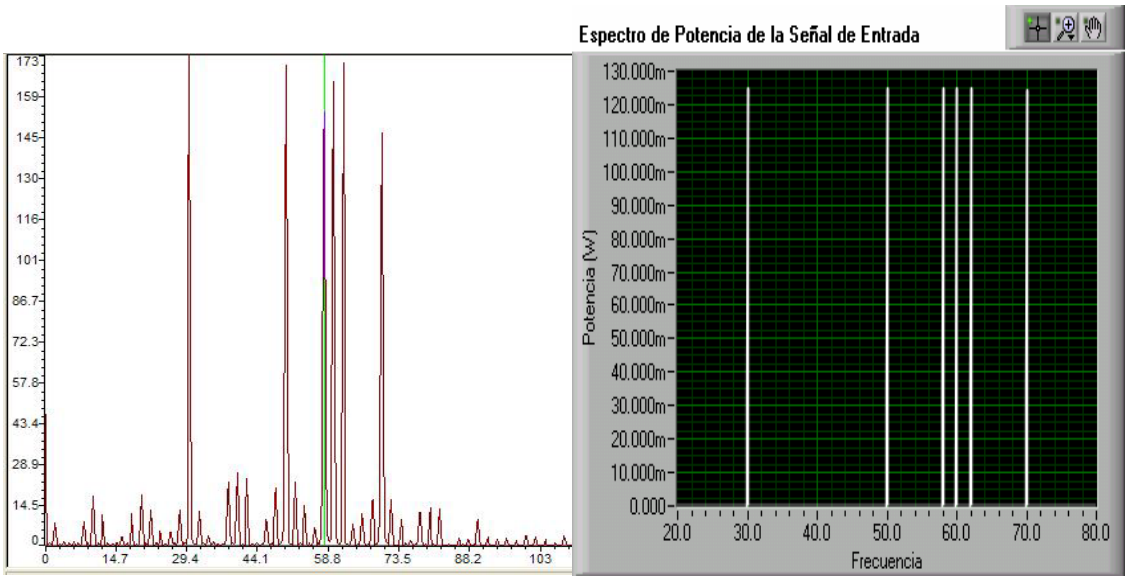


Figura 4.7 Espectro en frecuencia de la suma de 6 tonos.

4.2 PRUEBAS DE DISEÑO DE LOS FILTROS DIGITALES

- **Comparación de varias topologías de filtros IIR**

Para realizar esta prueba se establecieron las especificaciones observadas en la Figura 4.8 para un filtro pasa bajas IIR.

Figura 4.8 Configuración de las especificaciones de diseño IIR.

Tabla 4.1. Comparación de diferentes topologías IIR para las mismas especificaciones de un filtro pasa bajas.

Topología	Atenuación causada por el rizado en la banda de paso [dB]	Atenuación causada por el rizado en la banda de rechazo [dB]	Orden resultante
Elíptico	1	110	15
Chebyshev Inv.	-	110	49
Chebyshev	1	-	49
Butterworth	-	-	364
Bessel	-	-	364

De acuerdo a la Tabla 4.1 se puede concluir que para un grupo de especificaciones de diseño de un filtro IIR, la topología elíptica es la que puede cumplir esas especificaciones con el menor orden posible.

- **Comparaciones de varias topologías de filtros FIR**

Para realizar esta prueba se establecieron las especificaciones observadas en la Figura 4.9 para un filtro pasa bajas FIR.

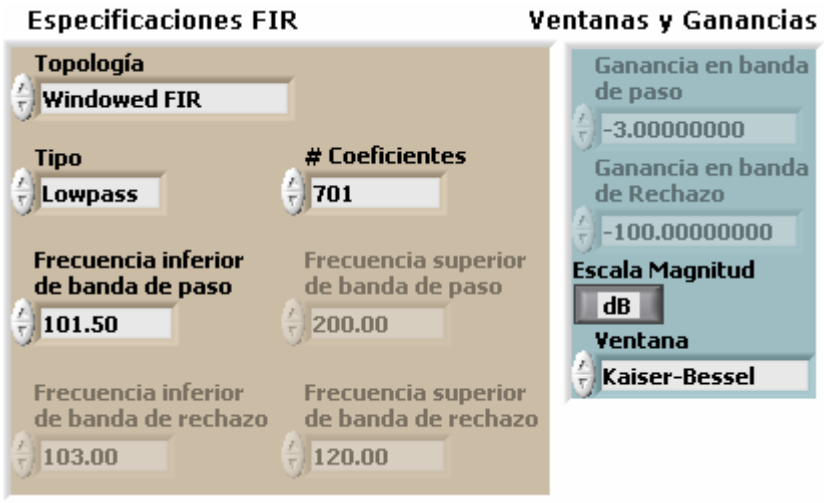


Figura 4.9 Configuración de las especificaciones de diseño FIR.

Para obtener resultados comparables entre la magnitud de la respuesta en frecuencia de los filtros IIR y los FIR, el valor de frecuencia de corte se fijó en 101.5Hz, que corresponde a un promedio entre las frecuencias de paso y rechazo definidas en el diseño IIR. Esta técnica se utiliza en el diseño de filtros FIR ventaneados.

Tabla 4.2. Comparación de diferentes topologías FIR para las mismas especificaciones de un filtro pasa bajas.

Topología	Tipo de ventana	Orden	A(dB) 100Hz	A(dB) 103Hz	Frecuencia de rechazo(Hz)	A(dB) frecuencia de rechazo
Ventaneado	Kaiser-Bessel	1000	0.45	45	105.5	87
	Hanning	1000	0.4	32.5	110	83
	Hamming	1000	0.35	38	104.5	60.25
	Triangular	1000	0.75	22.8	120	46
	Rectangular	1000	0.05	23.5	120	52.5
Rizado Constante ($\delta_1 = \delta_2$)	-	1000	0.28	48	104.5	56.75

A(dB): Valor de la Atenuación en decibeles

δ_1 : Máximo rizado en la banda de paso

δ_2 : Máximo rizado en la banda de rechazo

En estas pruebas, tal como se esperaba de la teoría, no se logró diseñar un filtro con el método de ventanas que logre cumplir las mismas especificaciones de los filtros IIR diseñados previamente, ya que para este método no es posible tener atenuaciones diferentes en las bandas de paso y rechazo. En el diseño con ventanas el orden se estima con la mayor atenuación, por ejemplo para la ventana Kaiser el orden es aproximadamente

$$M = \frac{A - 8}{2.285 \Delta w}$$

Para el caso de estas pruebas el orden sería

$$M = \frac{110 - 8}{2.285\left(\frac{2 * \pi * 3}{1000}\right)} = 2368$$

Para este orden la aplicación no arroja resultados consistentes, por tanto limitamos el orden a 1000, ya que con este orden los resultados son aceptables como se indica en la Tabla 4.2.

Con el diseño de rizado constante con mismos valores pico de rizado en la banda de paso y de rechazo se necesita un orden de aproximadamente 2214 según la Ecuación (1.114). Por tanto para tener resultados consistentes también se limitó el orden a 1000.

Con el diseño utilizando especificaciones se alcanzó una magnitud de respuesta en frecuencia que cumplen con especificaciones similares a las de los filtros IIR. De hecho como se observa en la Tabla 4.3 con un orden 622 se alcanzan los resultados esperados. Sin embargo fue necesario aumentar el ancho de banda de transición a 5Hz para obtener un orden con resultados consistentes en la aplicación. Para el diseño FIR el mayor orden calculado empíricamente que produce resultados consistentes es de 1000.

Tabla 4.3 Diseño FIR con Especificaciones

Tipo	Atenuación causada por rizado en banda paso	Atenuación causada por rizado en banda de arada	Orden
Especificaciones de Rizado constante $\delta_1 \neq \delta_2$	1	110	622

δ_1 : Máximo rizado en la banda de paso

δ_2 : Máximo rizado en la banda de rechazo

4.3 PRUEBAS DE ADQUISICIÓN Y FILTRADO EN TIEMPO REAL

Estas pruebas se componen de una parte de adquisición de una señal seguida por otra parte de filtrado en tiempo real de la señal adquirida. Para lograr esto se ejecutaron las aplicaciones de generación y filtrado simultáneamente de manera que la señal analógica generada se realimentaba a la TAD PCI 1200 para ser adquirida, procesada digitalmente y visualizada en el PC por la aplicación de filtrado.

4.3.1 Adquisición de Tres Tonos en 1Hz, 2Hz y 3Hz

Se generó una señal de fórmula: y(t)=sin(6.2831*t)+ sin(6.2831*2*t)+ sin(6.2831*3*t) [V] utilizando el generador implementado en este proyecto con una rata de generación de 1000 muestras/s y 2000 muestras. En la Figura 4.10 se observa la señal medida por el *FLUKE.*

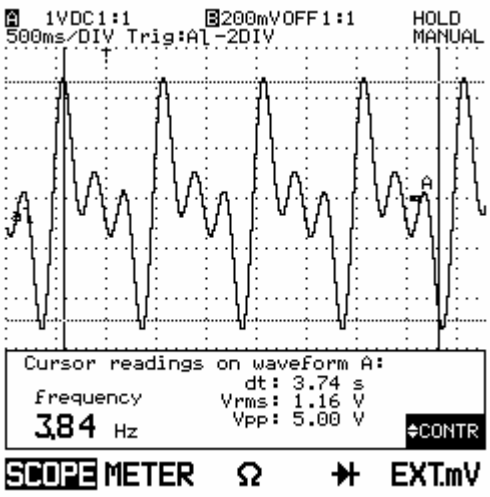
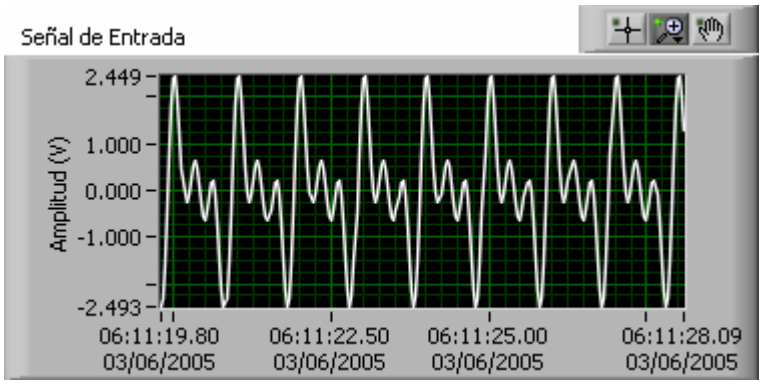
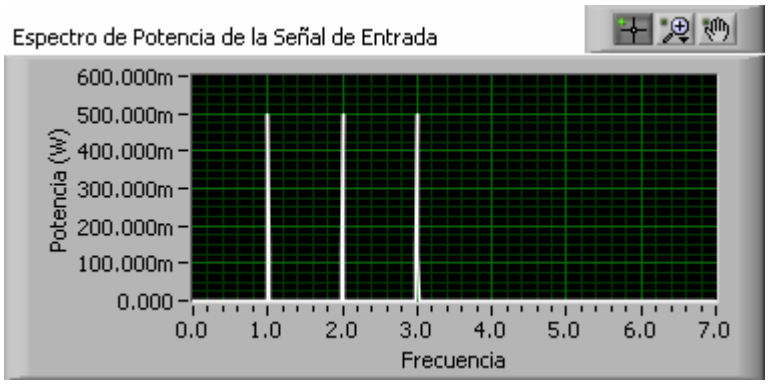


Figura 4.10 Suma de tres tonos senoidales.

La señal fue adquirida por la aplicación de filtrado con una frecuencia de muestreo de 30Hz y 30000 muestras. En la Figura 4.11a se muestra la señal de entrada y en la 4.11b su espectro de potencia. Ambas gráficas visualizadas por la aplicación de filtrado.



(a)



(b)

Figura 4.11 Suma de 3 tonos adquirida **a)** Señal de entrada adquirida por la TAD. **b)** Espectro de la señal de entrada.

4.3.2 Filtrado Pasa-Altas en Tiempo Real de la Suma de 3 Tonos Utilizando un Filtro IIR

- **Filtro Elíptico**

Para realizar esta prueba se diseñó un filtro elíptico pasa altas de orden 12 con frecuencia de corte en 2.5Hz para realizar el filtrado digital en tiempo real de la suma de tonos adquirida. Las especificaciones de este filtro y la magnitud de su respuesta en frecuencia se observan en la Figura 4.12.

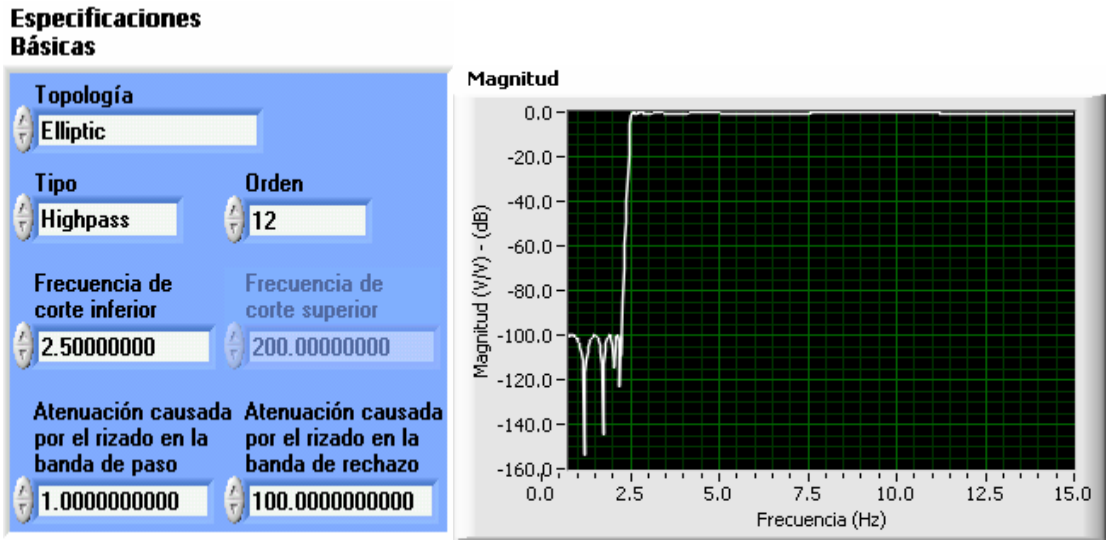


Figura 4.12 Filtro IIR elíptico pasa-altas.

En la Figura 4.13 se observa la señal de salida visualizada en el PC, la cual como se esperaba es una senoidal de aproximadamente 1V de amplitud. Se aprecia una pequeña atenuación en la amplitud de la señal debida al rizado en la banda de paso presente en los filtros elípticos. También se observa el espectro de potencia de la señal filtrada, un pulso de amplitud 0.4W ubicado en 3Hz.

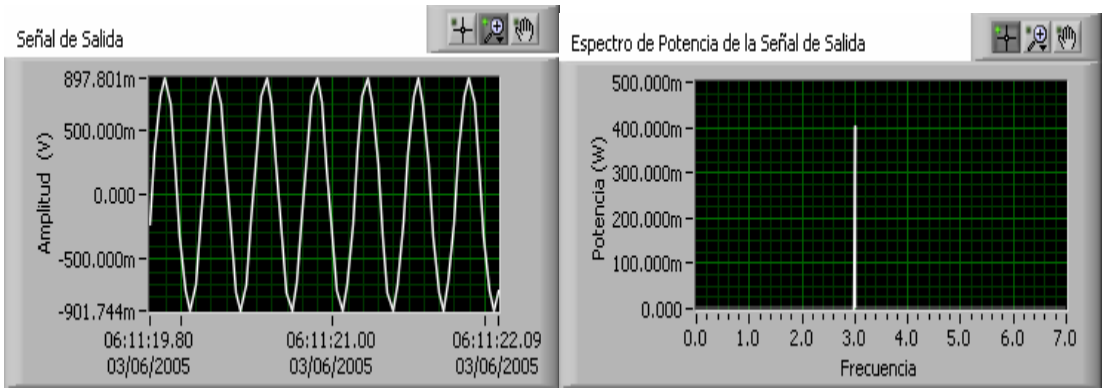


Figura 4.13 Señal de salida y su espectro luego de ser procesada por un filtro pasa-alto elípticas.

La señal de salida se guardó en un archivo de texto y fue cargada en el generador de señales arbitrarias para su posterior generación. En la Figura 4.14 se observa la señal de salida resultado del filtrado digital pasa altas.

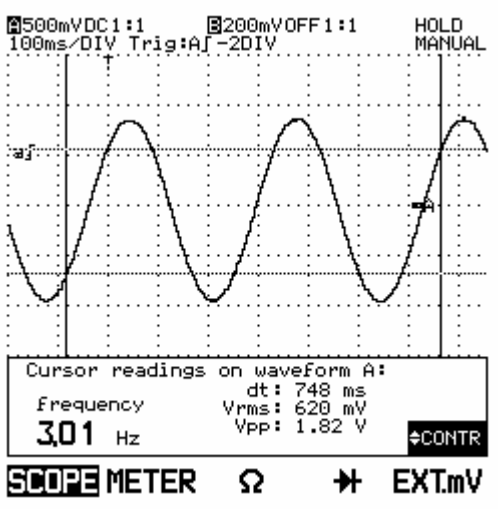


Figura 4.14 Señal de salida medida en el *FLUKE*

- **Filtro Butterworth**

Para realizar el filtrado pasa-altas también se diseñó un filtro Butterworth con las especificaciones y magnitud de respuesta en frecuencia observadas en la Figura 4.15.

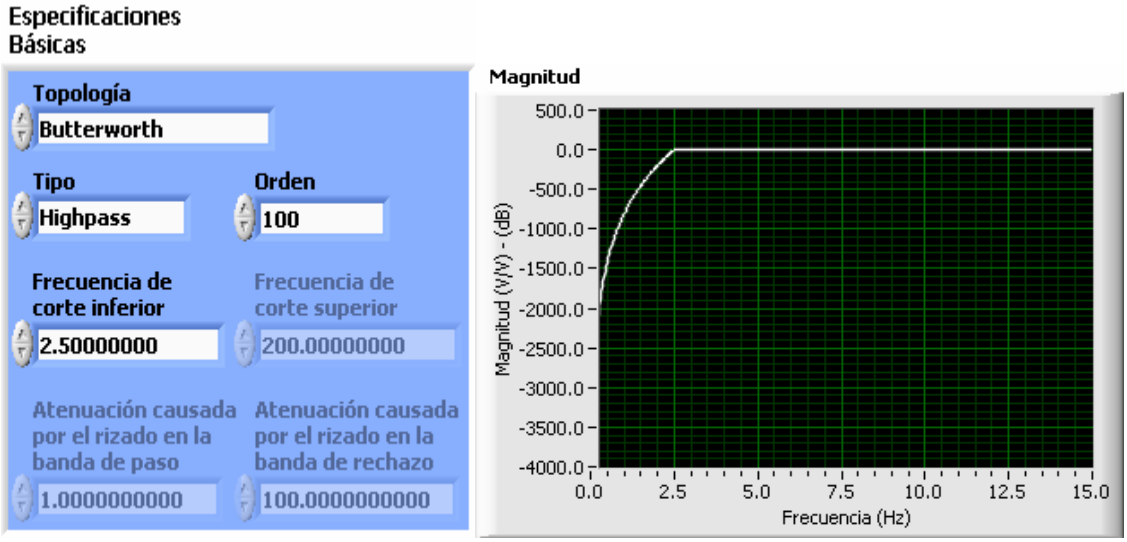


Figura 4.15 Filtro IIR Butterworth pasa-altas.

En la Figura 4.16 se observa la señal de salida, la cual como se esperaba es una senoidal de aproximadamente 1V de amplitud. .También se observa el espectro de potencia de la señal filtrada, un pulso de amplitud 0.5W ubicado en 3Hz.

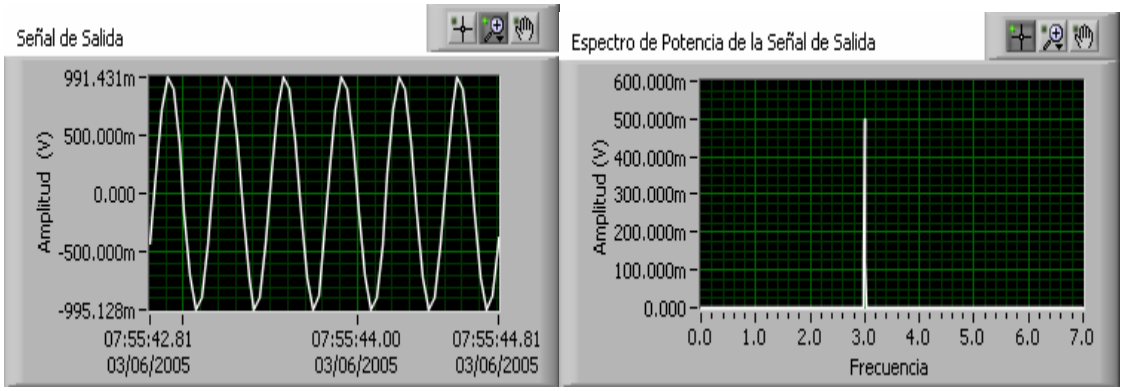


Figura 4.16 Señal de salida y su espectro de potencia luego de ser procesada por un filtro pasa-altas Butterworth.

Con el filtro pasa-altas Butterworth se necesitó un orden mucho mayor que con el filtro elíptico (100>>12) para eliminar los tonos en 1Hz y 2Hz. Sin embargo gracias a la respuesta máximamente plana del filtro Butterworth el tono de salida conserva prácticamente su potencia original, en contraste con la atenuación de 100mW producida por el rizado en la banda de paso del filtro elíptico.

- **Filtro FIR pasa-altas con ventana Kaiser**

El filtrado pasa-altas de los 3 tonos se realizó también con un filtro FIR diseñado con el método de ventanas. Se utilizó una ventana Kaiser. En la Figura 4.17 se muestra la magnitud de la respuesta en frecuencia de un filtro FIR pasa-altas de orden 50.

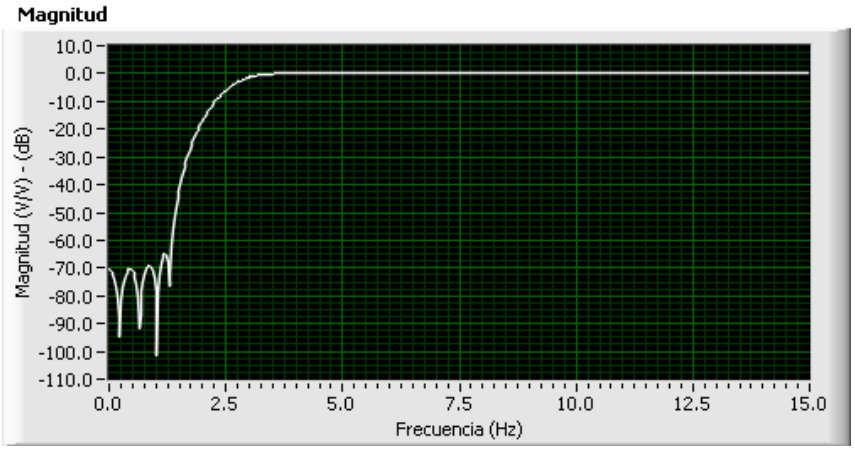


Figura 4.17 Filtro FIR pasa-altas. Ventana Kaiser. Orden 50. F_c=2.5Hz

Los resultados luego de procesada la señal digitalmente se observan en la Figura 4.18. Se observa que la señal de salida no representa exactamente el tono de 3Hz que se quería conservar. Gracias al espectro de potencia de la señal de salida se logra entender porque la señal no representa el tono, pues este fue atenuado más de 100mW y además el tono en 2Hz no fue totalmente eliminado y por tanto esta componente todavía aparece en la señal de salida visualizada en el PC.

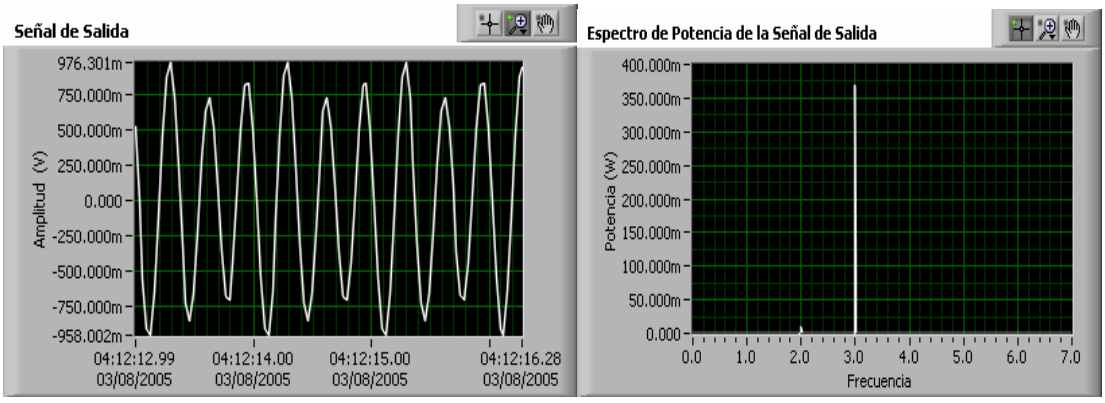


Figura 4.18 Señal de salida y su espectro de potencia luego de un filtrado pasa-altas con un filtro FIR de orden 50.

Para mejorar este resultado se aumentó el orden del filtro FIR pasa-altas a 100. En la Figura 4.19 se observa la magnitud de la respuesta en frecuencia de este filtro.

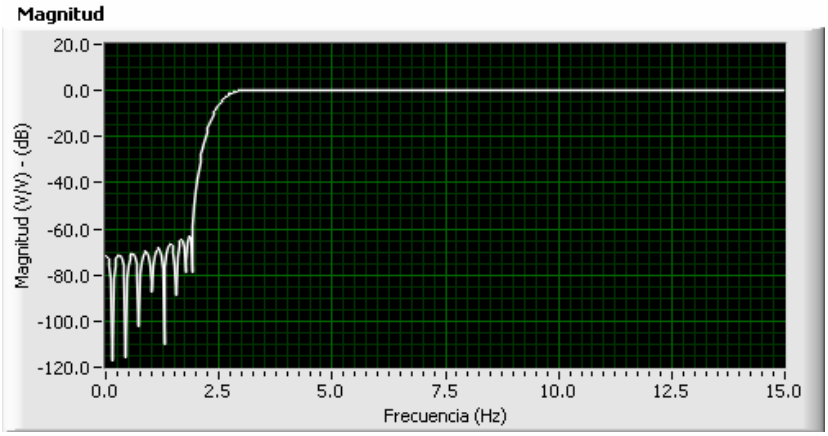


Figura 4.19 Filtro FIR pasa-altas. Ventana Kaiser. Orden 100. Fc=2.5Hz

Los resultados de procesar la suma de 3 tonos con este filtro se observan en la Figura 4.20. Se observa la mejora en el tono de salida resultante, el cual tiene una amplitud de aproximadamente 1V. El espectro en potencia de la señal de salida muestra como se conserva la potencia de 500mW del tono de 3Hz y las componentes en 1Hz y 2Hz han sido atenuadas de manera efectiva, y gracias a esto no tienen efecto apreciable en la señal de salida.

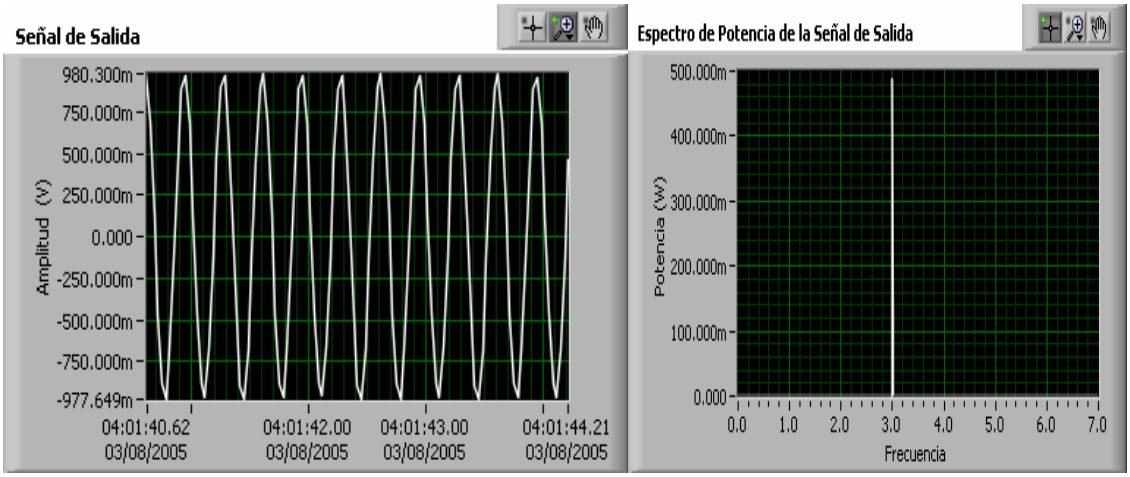


Figura 4.20 Señal de salida y su espectro de potencia luego de un filtrado pasa-altas con un filtro FIR de orden 100.

4.3.3 Filtrado Para-Banda en Tiempo Real de la Suma de Tres Tonos Utilizando un Filtro FIR

Para realizar esta prueba se diseñó un filtro IIR para banda. Se utilizó una ventana Hanning de 201 coeficientes. La frecuencia central del filtro se estableció en 2Hz. Las especificaciones de este filtro y la magnitud de su respuesta en frecuencia se observan en la Figura 4.21.

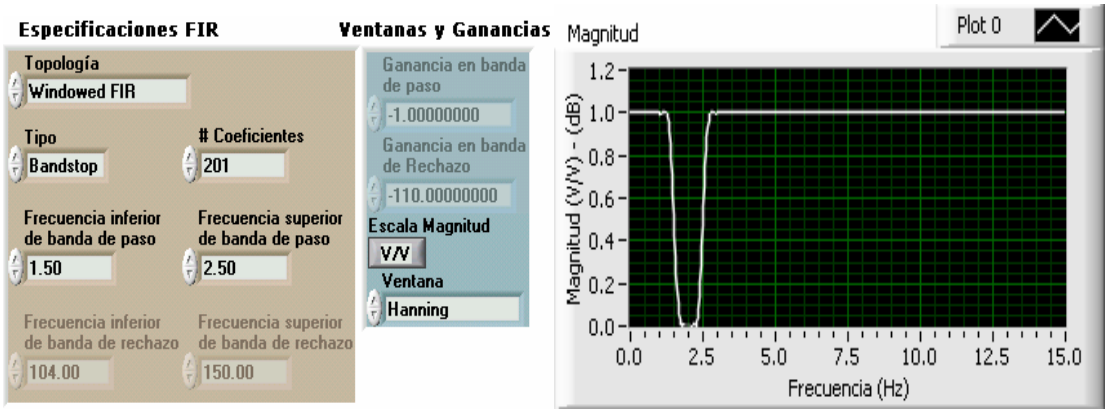


Figura 4.21 Filtro FIR ventaneado para-banda.

En la Figura 4.22a y 4.22b se observa la señal de salida y su espectro de potencia respectivamente. Claramente se observa la eliminación del tono ubicado en 2Hz. Además no hay atenuación apreciable en los tonos que se deseaban conservar en 1Hz y 2Hz.

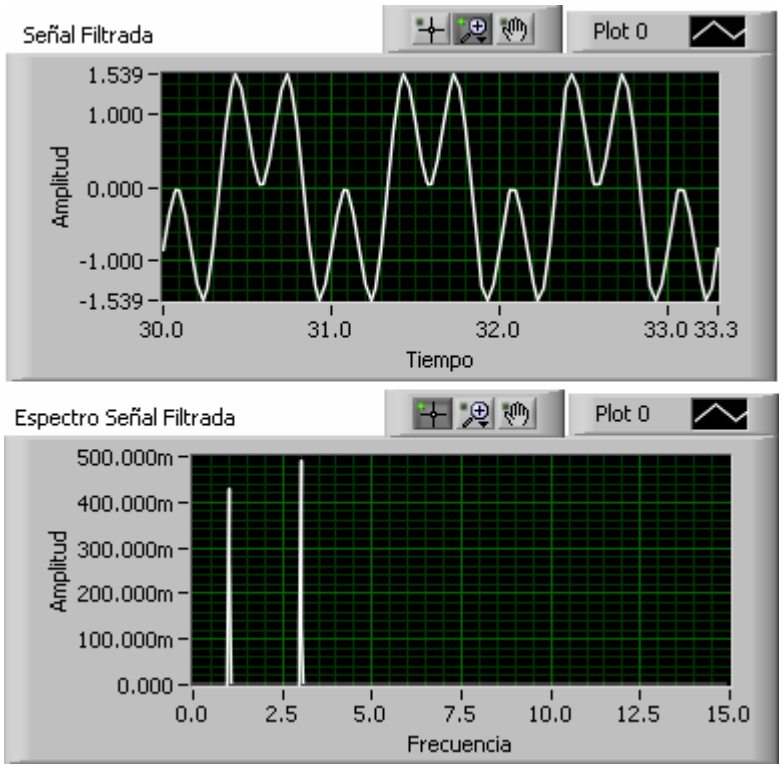


Figura 4.22 Señal procesada por el filtro para banda **a)** Señal de salida del filtro para banda. **b)** Espectro de potencia de la señal de salida

La señal de salida se guardó en una archivo de texto para luego ser cargada en el generador de señales arbitrarias. La Figura 4.23 es la señal de salida medida en el *FLUKE.*

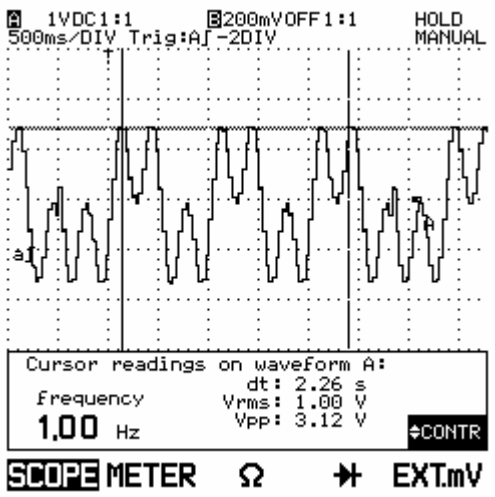


Figura 4.23 Señal de salida medida en el *FLUKE.*

4.3.4 Adquisición de una Señal Cuadrada

Para realizar esta prueba se generó una señal cuadrada 1V de amplitud y 10Hz de frecuencia utilizando el generador implementado en este proyecto. En la Figura 4.24 se observa la señal medida en el *FLUKE.*

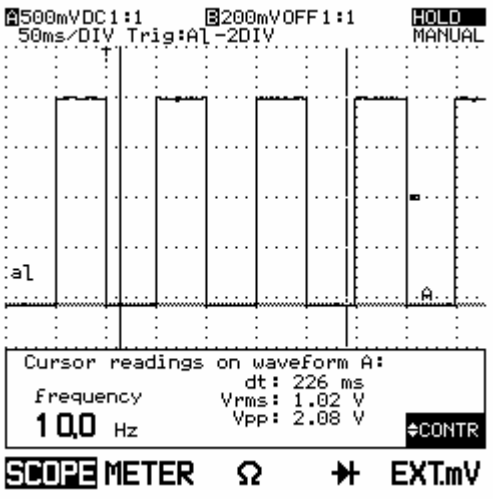
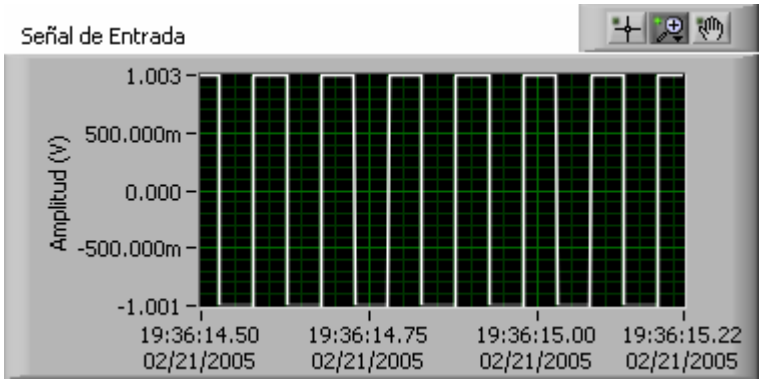
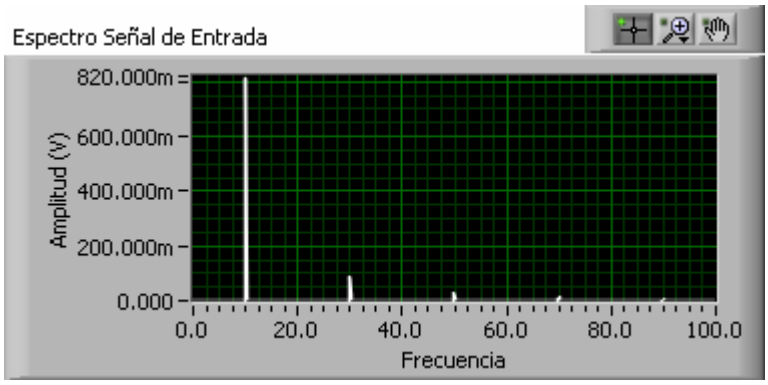


Figura 4.24 Señal cuadrada medida en el *FLUKE*

La seña fue adquirida por la TAD con una frecuencia de muestreo de 1kHz y un número de muestras de 10000. La señal adquirida y su espectro de potencia se observan en la Figura 4.25a y 4.25b respectivamente.



(a)



(b)

Figura 4.25. Adquisición de señal cuadrada **a)** Señal cuadrada adquirida. **b)** Espectro de potencia de la señal cuadrada

4.3.5 Filtrado Pasa-Bajas en Tiempo Real de la Señal Cuadrada Utilizando un Filtro IIR

Para realizar esta prueba se diseñó un filtro pasa-baja elíptico de orden 23 y frecuencia de corte 20Hz. En la Figura 4.26 se pueden observar las especificaciones del filtro y la magnitud de su respuesta en frecuencia.

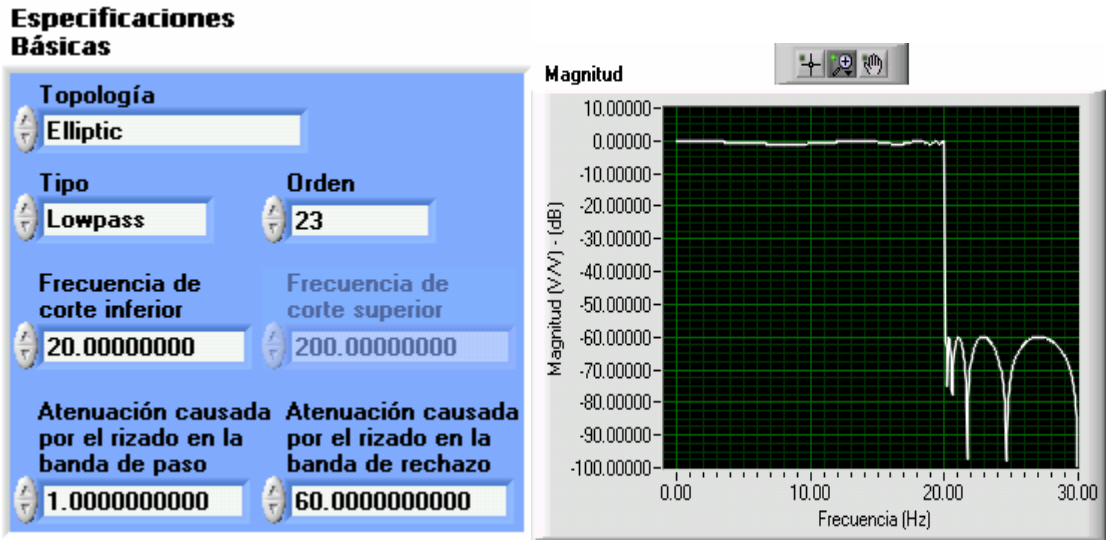


Figura 4.26 Filtro pasa-bajas elíptico.

En la Figura 4.27a y 4.27b se observa la señal de salida y su espectro de potencia respectivamente. Gracias a este filtrado se logró conservar la componente fundamental de la señal cuadrada ubicada en 10Hz. Se observa como los armónicos de la señal cuadrada fueron rechazados totalmente por el filtro pasa-bajas.

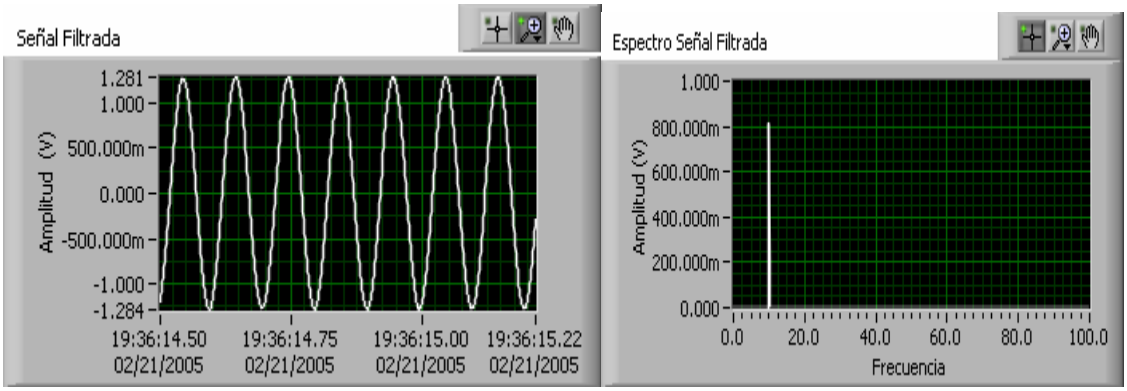


Figura 4.27 Señal procesada por el filtro pasa bajas **a)** Señal de salida luego del filtrado pasa-bajas. **b)** Espectro de potencia de la señal de salida luego del filtrado.

La señal de salida se guardó en un archivo de texto para luego ser cargada en el generador de señales arbitrarias. La Figura 4.28 es la señal de salida medida en el *FLUKE.*

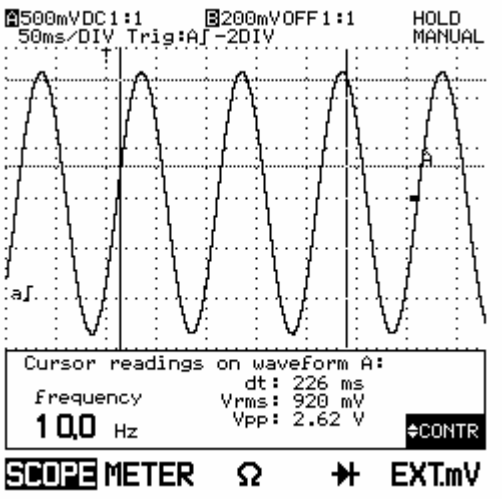


Figura 4.28 Señal de salida medida en el FLUKE.

4.3.6 Eliminación de Componente de 60Hz

Para esta prueba se generó una señal de fórmula: y(t)= 0.5*sin(6.2831*30*t)+0.5* sin(6.2831*50*t)+ 0.5*sin(6.2831*58*t)+0.5*sin(6.2831*60*t)+ 0.5*sin(6.2831*62*t)+ 0.5*sin(6.2831*70*t) [V] utilizando el generador implementado en este proyecto con una rata de generación de 2000 muestras/s y 5000 muestras.

Simultáneamente se adquirió la señal utilizando la aplicación de filtrado digital en tiempo real con una frecuencia de muestreo de 1kHz y 50000 muestras. En la Figura 4.29 se muestra la señal adquirida y su espectro de potencia visualizados en el PC.

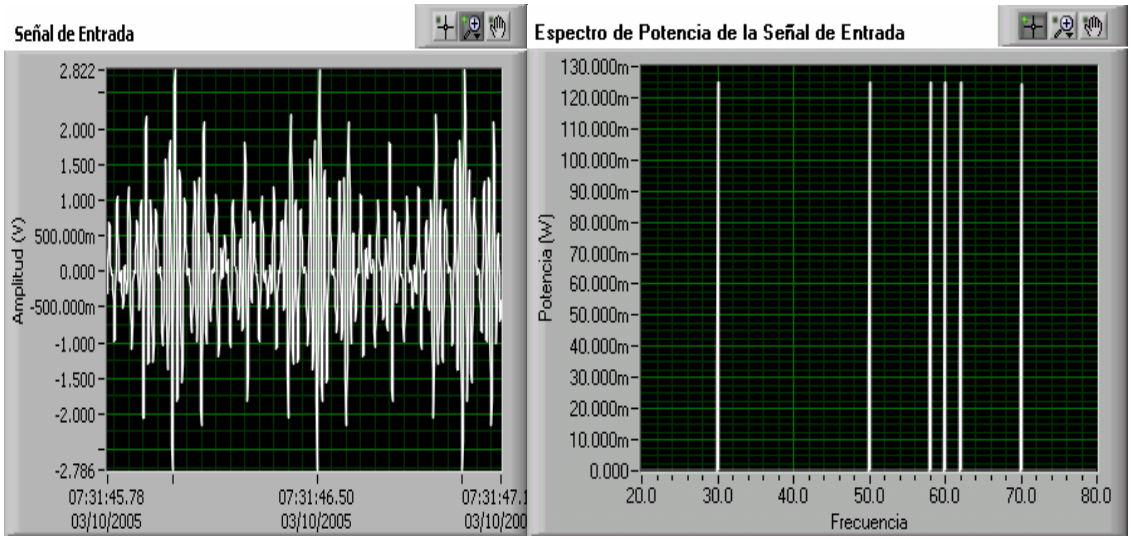


Figura 4.29 Señal adquirida y su espectro de potencia.

Se procedió a diseñar un filtro notch digital para eliminar la componente de 60Hz. Se escogió un filtro IIR con topología elíptica de orden 20. El filtro se diseñó con las especificaciones mostradas en la Figura 4.30. También se observa en la figura la magnitud de la respuesta en frecuencia resultante.

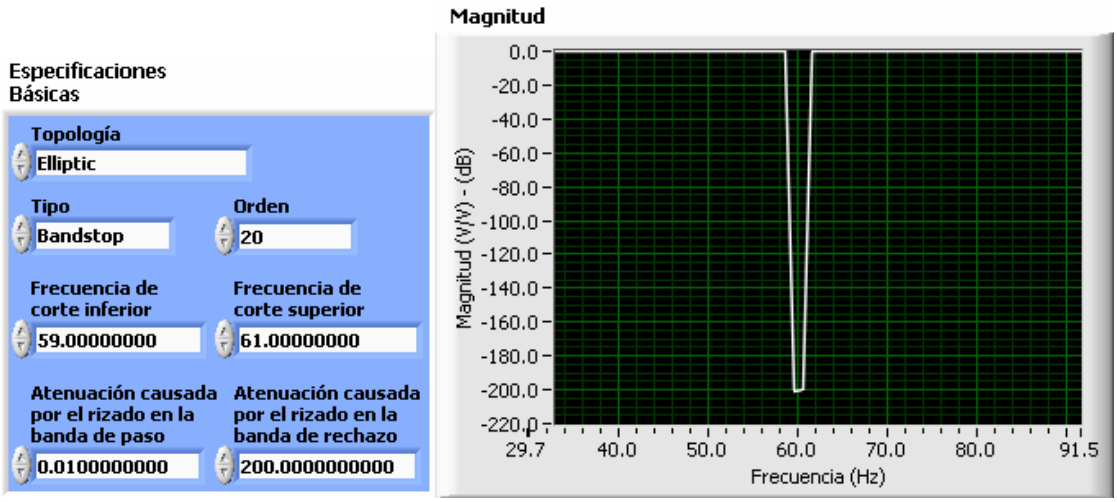


Figura 4.30 Especificaciones y magnitud del filtro elíptico.

Luego del filtrado en tiempo real se obtuvieron los resultados observados en la Figura 4.31. Se aprecia claramente la eliminación de la componente de 60Hz. Todas las demás componentes se conservaron prácticamente intactas gracias a la alta selectividad del filtro diseñado.

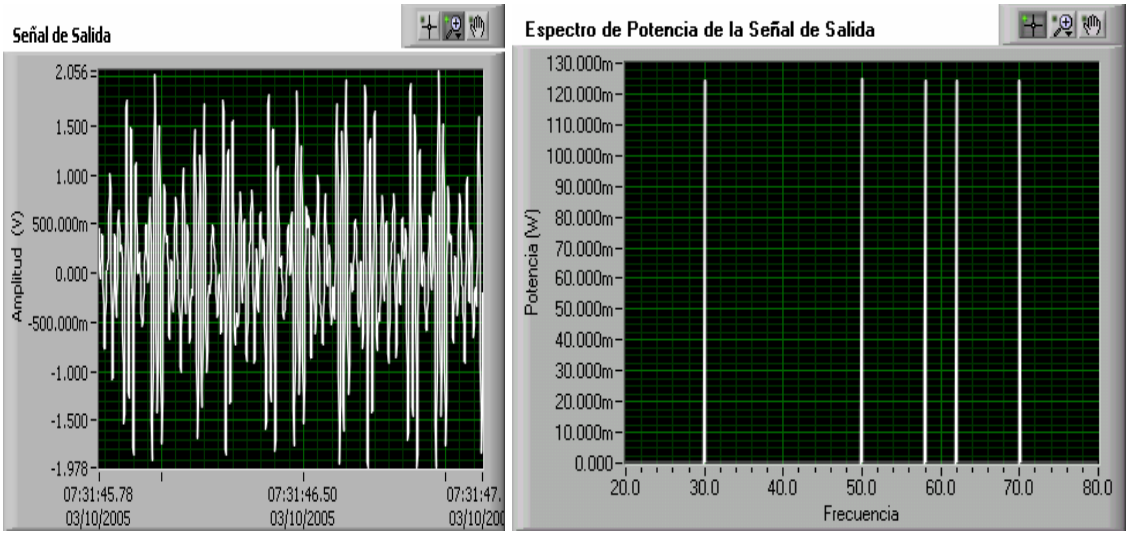


Figura 4.31 Señal de salida y su espectro en potencia.

CONCLUSIONES

En este capitulo se recopilan los logros y observaciones más relevantes obtenidas a lo largo del desarrollo de las etapas necesarias para cumplir los objetivos propuestos.

Se diseñó un algoritmo en el lenguaje de programación gráfica LabVIEW con el fin de realizar un filtrado digital en tiempo real a una señal analógica y visualizar en un computador la señal procesada y su espectro de potencia, utilizando como dispositivo de adquisición de datos una tarjeta PCI 1200 de *National Instruments.*

Este software se puede explotar de forma didáctica para la comprensión de los conceptos básicos del procesamiento digital. En la adquisición de la señal analógica se pueden observar los efectos del muestreo al variar parámetros como frecuencia de muestreo, número de muestras y el uso de *buffers* circulares. En el filtrado digital se pueden simular una gran variedad de filtros con topologías integradas en LabVIEW o ingresando los coeficientes de la función de transferencia y observar los cambios en su respuesta en frecuencia, tanto en la magnitud como en la fase. También permite el tratamiento digital de señales analógicas, en modo de simulación o en tiempo real, dando la posibilidad al usuario de manipular los parámetros más significativos de los filtros como son los rizados, las frecuencias de corte y ganancias en las bandas de paso y rechazo. El modo de simulación permite evaluar la respuesta de un filtro digital a una señal de entrada simulada en el PC , lo cual permite predecir su salida sin necesidad de implementar el sistema real y así evitar los errores de diseño y pérdidas económicas causadas por implementar un sistema sin haber realizado una simulación previa. El modo en tiempo real permite introducir al PC una señal analógica, para ser procesada digitalmente por cualquier tipo de filtro programado o cargado en la aplicación de filtrado de forma tal que la salida del filtro se puede observar en tiempo real en el PC. Este modo de operación es de gran utilidad para señales que varían su frecuencia y amplitud de manera irregular en el tiempo, como es el ejemplo de una señal electrocardiográfica.

Se implementó un generador de señales arbitrarias basado en el hardware de una tarjeta de adquisición, el computador personal y el software de programación gráfica LabVIEW. Este generador de señales demostró ser de gran utilidad como herramienta de laboratorio para evaluar filtros digitales. Su utilidad puede extenderse a aplicaciones analógicas, como generar armónicos, estimular circuitos electrónicos o cargar señales de bases de datos adquiridas en otras aplicaciones.

Trabajando en conjunto con la aplicación de generación y la aplicación de filtrado se logró evaluar el comportamiento de las diferentes familias de filtros ante diferentes señales de entrada y compararlas entre sí. Se destaca que los filtros IIR alcanzan prestaciones mucho más exigentes que los filtros FIR con un orden mucho menor, lo que a la su vez implica un tiempo de procesamiento también menor. Sin embargo, los filtros IIR tienen la desventaja de una fase no lineal y por tanto los filtros FIR son atractivos cuando la aplicación exige una distorsión de fase nula.

Los filtros digitales estudiados en este proyecto mostraron grandes ventajas respecto a su contraparte analógica. La experiencia con los filtros analógicos adquirida durante la carrera, hace recordar la dificultad para diseñar filtros de orden alto y pendientes pronunciadas en la banda de transición debido a la necesidad de utilizar componentes electrónicos como resistencias y condensadores que presentan tolerancias altas y es dificil conseguir los valores exactos necesarios para alcanzar una frecuencia de corte específica. Además la teoría analógica de filtros implica conocimientos matemáticos de derivadas e integrales en caso de que se quiera analizar la respuesta en frecuencia de un sistema expresado por una ecuación diferencial. Algo muy diferente sucedió con los filtros digitales, los cuales demostraron ser muy flexibles, inmunes al ruido pues están implementados por software y con respuestas en frecuencia que permiten realizar procesamientos muy exigentes. Además ya que se trabaja con secuencias de muestras y operaciones aritméticas básicas de multiplicación y adición, las respuestas de los sistemas discretos definidos por ecuaciones en diferencias con coeficientes constantes pueden ser verificadas matemáticamente de una forma rápida. Para el análisis espectral se utilizaron técnicas exactas, como la FFT, que permiten a LabVIEW aproximar los

espectros de las señales muestradas de forma que este análisis no ocupe el procesador de manera significativa.

Las dos aplicaciones implementadas en este proyecto pueden trabajar en conjunto para servir de herramienta en investigaciones, como por ejemplo en el campo de la bioingenieria, donde es muy importante reproducir en cualquier momento una señal a partir de sus muestras adquiridas y almacenadas previamente para su posterior estudio. Es el caso del estudio de arritmias en electrocardiogramas, donde con la aplicación de filtrado se puede guardar en un archivo de texto un tiempo prolongado de la señal de entrada adquirida en tiempo real y con el generador de señales arbitrarias se podría reproducir en cualquier momento esta secuencia de muestras que con la información correcta de tiempo representaría la señal adquirida en un tiempo pasado y que puede ser estudiada y analizada en un momento más favorable.

RECOMENDACIONES

El estudio del procesamiento digital de señales es un campo poco conocido por estudiantes de pregrado en la escuela de ingenierías eléctrica, electrónica y de telecomunicaciones; por lo tanto se recomienda el uso de las aplicaciones desarrolladas en este proyecto como herramientas poderosas para la investigación de sistemas discretos ya que permiten la simulación con un alto grado de confiabilidad y a un bajo costo. Esto se podría lograr al añadir la aplicación **Simulación.vi** que se encuentra almacenada en el CD que acompaña este proyecto, a las prácticas de laboratorio de una asignatura de tratamiento digital de señales con el fin de apoyar didácticamente la comprensión de los filtros digitales, promoviendo de esta forma futuras investigaciones en este campo.

Se recomienda el uso de las herramientas desarrolladas en este proyecto como apoyo de futuros proyectos relacionados con la Bioingeniería, ya que se pueden generar señales previamente adquiridas y guardadas y de esta forma se permite el estudio de dichas señales en un ambiente de laboratorio e incluso su procesamiento discreto.

Se recomienda que la universidad enfoque los proyectos orientados a la generación de señales a desarrollos en *software,* debido a su versatilidad y buenos resultados.

En proyectos futuros se recomienda la implementación de un sistema de procesamiento discreto que realice el proceso completo en tiempo real, desde la adquisición de la señal analógica, pasando por el procesamiento en tiempo discreto y terminando con la reconstrucción de la señal procesada. Para lograr esto en tiempo real se recomienda utilizar un DSP ya que es un procesador diseñado específicamente para el procesamiento digital de señales y logra prestaciones mucho más altas que la tarjeta de adquisición PCI 1200 ya que esta tarjeta a pesar de ser de alto desempeño es una tarjeta de bajo costo que difícilmente logra aproximarse a una aplicación en tiempo real.

BIBLIOGRAFÍA

[1] OPPENHEIM, Alan V. y WILLSKY Alan S. *Señales y Sistemas.* 2da Edición. Prentice Hall, 1998.

[2] OPPENHEIM, Alan V. *Tratamiento de Señales discretas.* Segunda edición. Editorial Prentice Hall Iberia, Madrid, 2000.

[3] PROAKIS, J.G. y MANOLAKIS, C.T. *Digital Signal Processing Principles, Algorithms, and Applications*. 3ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

[4] NATIONAL INSTRUMENTS. *LabVIEW Printed Manuals.* Austin, Texas, Julio 2000.

[5] MATHWORKS, *Signal Processing Toolbox Users Guide,* The Mathworks, Inc. Natick, MA, USA, 1998.

[6] XIAOHUI XU y BEHROUZ NOWROUZIAN. *Design of FIR digital filters using weighted peak-constrain.* Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering. Shaw Conference Center, Edmonton, Alberta. Cánada Mayo 9-12 1999, p.805-810.|

[7] ARGENTI, Fabrizio y DEL RE, Enrico. *Design of IIR Eigenfilters in the Frequency Domain.* IEEE Transactions On Signal Processing, Vol. 46, No. 6, p. 1964-1968, Junio 1998.

[8] ICONTEC. *Norma técnica colombiana para trabajos escritos.*
Febrero 2004.

ANEXOS

ANEXO A: DESCRIPCIÓN DE LAS APLICACIONES UTILIZADAS EN LABVIEW

INTRODUCCIÓN A LA PROGRAMACIÓN EN LABVIEW

LabVIEW es un lenguaje de programación gráfico que utiliza iconos en lugar de líneas de texto para crear aplicaciones. En contraste con los lenguajes de programación basados en líneas de código, donde las instrucciones determinan la ejecución del programa, LabVIEW usa programación por flujo de datos, donde el flujo de los datos determina la ejecución.

Instrumentos Virtuales

Los programas diseñados en LabVIEW se conocen como Instrumentos Virtuales (VIs por sus siglas en inglés) porque su operación y apariencia imitan a los instrumentos físicos convencionales como los generadores de señales, osciloscopios o multímetros.

Un VI está compuesto de los siguientes elementos:

- **Panel frontal:** Es la interfaz gráfica con el usuario. Se construye con controles e indicadores, los cuales son las entradas y salidas del VI respectivamente. Los controles pueden ser pulsadores, switches, perillas u otro control. Los indicadores pueden ser LEDs, gráficas u otro indicador. Los controles simulan las entradas de datos a los instrumentos físicos y los indicadores simulan las salidas.

- **Diagrama de Bloques**: Contiene el código gráfico que define la funcionalidad del VI. Este código se añade luego de construir el panel frontal ya que los controles e indicadores aparecen como variables dentro del diagrama de bloques y deben relacionarse de acuerdo al diseño y funcionalidad del VI.

- **Icono y panel de conectores**: Identifican al VI de forma que pueda ser utilizado y cableado dentro de otro VI. Un VI dentro de otro VI se define como un subVI. Es lo mismo que los subprogramas o subrutinas programadas en código de líneas de instrucciones.

ESTRUCTURAS DE CASOS Y CICLOS EN LABVIEW

Las estructuras en LabVIEW son representaciones gráficas de los comandos de casos y ciclos de los lenguajes basados en líneas de código. Se utilizan para repetir una sección del diagrama de bloques un determinado número de veces o para ejecutar un bloque del diagrama cuando se cumple una condición específica.

- **Ciclo While:** Ejecuta un subdiagrama hasta que se cumpla cierta condición. Es similar a los comandos Repeat Until de los programas basados en líneas de código.

- **Ciclo For:** Ejecuta un subdiagrama un número de veces establecido. Tiene una entrada de ***N*** para establecer el número de iteraciones, y una salida ***i*** que representa el número de iteraciones realizadas.

- **Estructura de Caso:** Contiene una serie de subdiagramas, de los cuales solo uno se ejecuta dependiendo del caso seleccionado a la entrada de la estructura. La estructura ejecuta un solo caso a la vez.

- **Estructura de Secuencia:** Esta estructra contiene una serie de subdiagramas que se ejecutan en un orden secuencial. Esta es una forma fácil de asegurar que una acción determinada dentro del programa se ejecute antes o después de otra. La estructura no retorna sus datos hasta no terminar la última acción.

ALMACENAMIENTO DE DATOS EN LABVIEW

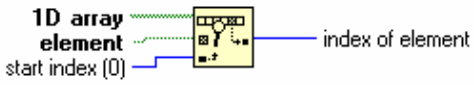
Para almacenar datos de formas de onda en un *buffer*, almacenar datos de una magnitud de respuesta en frecuencia de un filtro o para agrupar una serie de controles de diferente tipo, LabVIEW utiliza dos herramientas principales: los arreglos y los clusters.

Arreglo: Un arreglo está compuesto de elementos y dimensiones. Los elementos conforman los datos del arreglo. Las dimensiones son la longitud, altura o profundidad del arreglo. En este proyecto se utilizaron arreglos de máximo dos dimensiones, por tanto se puede hacer una analogía con matrices, donde una dimensión son las filas y la otra las columnas. Los arreglos se utilizan para almacenar datos del mismo tipo. Son ideales para realizar operaciones repetitivas y también para realizar operaciones aritméticas entre ellos.

Cluster: A diferencia de los arreglos, los clusters pueden agrupar datos de diferente tipo. Este agrupamiento facilita el cableado ya que un solo cluster puede contener una gran cantidad de controles o indicadores que se podrían enviar a un subVI por un solo conectar y por tanto utilizar una sola entrada del subVI. Al igual que un arreglo, el cluster tiene un orden específico dentro de sus elementos. Sin embargo para utilizar algún elemento del cluster deben desligarse todos los elementos que componen el cluster.

OTROS VIs UTILIZADOS EN EL PROYECTO

Search 1D Array

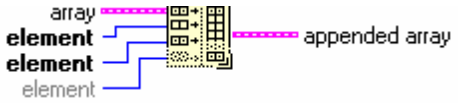


Este VI busca por el elemento en su entrada **element** dentro de un arreglo de una dimensión comenzando en el índice en su entrada **Start Index.** Las entradas **Element** y **Start Index** deben ser escalares. El índice de inicio en **Start Index** es 0 por defecto si no se usa. Su salida **index of element** es el índice donde se encontró el elemento y si no se encuentra ningún elemento su salida es -1.

Size(s)

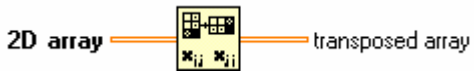
Esta aplicación retorna el número de elementos en cada dimensión del arreglo de entrada.

Build Array



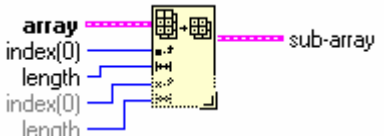
Construye un arreglo de n dimensiones a partir de arreglos de entrada de n-1 dimensiones. Si todas las entradas tienen la misma dimensión se pueden concatenar para formar un arreglo más grande de la misma dimensión.

Transpose 2D Array



Reorganiza el arreglo de entrada de dos dimensiones [i,j] de tal forma que a la salida se obtenga el arreglo transpuesto de dos dimensiones [j,i].

Array Subset



Retorna una porción de un arreglo comenzando en un índice y un número de elementos establecidos en sus entradas.

ANEXO B: GUÍAS DE USUARIO

Hoy día cualquier software posee una ayuda o por lo menos un manual de instrucciones que le indique al usuario cómo debe operar el programa. Por lo anterior, en este anexo se explica al usuario como debe utilizar las aplicaciones implementadas en este proyecto: el generador de señales arbitrarias y la aplicación de filtrado digital en tiempo real. Además se puede dar uso de la ayuda contextual de LabView para conocer los detalles de los controles e indicadores que componen los paneles frontales que se utilizan en esta aplicación.

Ambas aplicaciones necesitan los siguientes requerimientos mínimos:

Requerimientos mínimos: Computador personal con procesador Intel Pentium a 200 MHz o equivalente, 32 MB de RAM, monitor con configuración de color de 256 colores (se recomienda color de 16 bits), resolución de pantalla de 1024 x 768, Windows 98 o superior. Además, el PC debe tener instalado el software LabVIEW versión 6.i o superior.

Además se necesitan los archivos propios de las aplicaciones los cuales están guardados en un CD incluido con este proyecto de grado.

B.1 GENERADOR ARBITRARIO DE SEÑALES

Esta guía se hace con el fin de dar al usuario del generador de señales arbitrarias una metodología para operarlo de manera correcta, y para comprender en forma general el funcionamiento del software y hardware, para así explotar al máximo la funcionalidad del generador.

En las siguiente secciones se guiará al usuario sobre como se recomienda editar las formas de onda para que su experiencia con el generador sea eficiente y así lograr la satisfacción de obtener los resultados esperados.

Antes de Comenzar: Ubique el archivo *Generador Arbitrario.VI* y ejecútelo. Revise los requerimientos mínimos para la correcta ejecución del programa. Debe ubicar también los siguientes archivos que serán llamados durante la ejecución del programa: *Libreria.VI*, *Editar Form.VI* y *Mouse.VI*.

COMO UTILIZAR EL PANEL FRONTAL

Para una correcta edición de una forma de onda es necesario seguir los siguientes pasos:

1. Configurar el generador.
2. Seleccionar la Opción.
3. Pre-visualizar la forma de onda.
4. Iniciar la generación.

A continuación se explica detalladamente como realizar cada paso.

1. CONFIGURAR EL GENERADOR.

Este paso configura dos aspectos claves de la señal: resolución horizontal de la forma de onda y la frecuencia de muestreo.

La resolución horizontal de la forma de onda se establece con el control de ***Tamaño del buffer*** *(número de muestras)*. En la Figura B.8 se observa donde se encuentra este control. Este control se puede variar de dos formas: desplazando la barra azul hacia la

derecha con lo cual se aumenta el número de muestras o utilizando el control digital. El número de muestras debe ser entero. Por defecto se encuentra en 1000 muestras.

La ***Rata de generación*** es el número de muestras a generar por segundo. Por defecto se encuentra en 1000 muestras/seg. Se varía con el control indicado en la Figura B.8.

El usuario debe comprender claramente la relación entre la rata de generación y el número de muestras con la información de tiempo de la señal para editar correctamente la forma de onda. La relación se puede expresar por Ecuación (B.1):

$$Duración\ de\ un\ buffer\ de\ salida = (\frac{1}{rata\ de\ generación}) * (número\ de\ muestras) \quad [seg]$$

(B.1)

En otras palabras el resultado de la Ecuación B.1 es el tiempo que tarda la TAD PCI 1200 en generar todas las muestras de un *buffer* completo por el canal de salida seleccionado.

De este resultado podemos sacar las siguientes conclusiones:

- La rata de generación es en realidad una frecuencia de muestreo pues define un período entre muestras generadas, o sea la duración de tiempo entre muestras consecutivas. Este período de generación se define entonces con la Ecuación (B.2),

$$T_g = \frac{1}{rata.de.generación}[seg] \qquad (B.2)$$

- Si el usuario desea editar varios ciclos de una señal periódica dentro de un *buffer* de salida, debe calcular aproximadamente cuantas muestras representan un ciclo de la señal con la rata de generación seleccionada. Este calculo se puede realizar con la Ecuación (B.3).

$$\#\ \ de\,muestras\ por\,ciclo = \frac{rata\,de\,generación(muestras/seg)}{frecuencia\,señal\,deseada(ciclos/seg)} \qquad \left[muestras/ciclo\right]$$

(B.3)

Lo anterior se puede observar mejor en las figuras que se muestran a continuación. En la Figura B.1. se observan las muestras generadas para una señal senoidal de 1Hz con rata de generación de 10 muestras/seg y tamaño de *buffer* de 10 muestras. Es claro que la duración entre muestras consecutivas es T_g=1/10=100ms. La duración del *buffer* es por tanto (100ms)*(10muestras)=1s. La Figura B.2. representa la misma señal aumentando solamente el número de muestras a 20 con lo cual la duración del *buffer* aumenta a 2s pero la frecuencia de salida de la señal no cambia. La Figura B.3 representa la misma señal aumentando la rata de generación a 20 muestras/seg y 20 muestras por *buffer*. El usuario debe observar que la señal siempre es la misma sin embargo la señal de la Figura B.3 es una mejor reconstrucción de una señal senoidal analógica de 1Hz ya que tiene una mejor resolución horizontal y vertical que las otras 2 figuras pues tiene 20 muestras espaciadas tan solo 50 ms una de la otra y por tanto la interpolación entre ellas es menos abrupta que para las otras configuraciones. Esto se puede observar en las Figuras B.4, B.5 y B.6 donde las muestras se interpolan mediante un retenedor de orden cero.

Una regla práctica para obtener información de frecuencia confiable es configurar una rata de generación 10 a 50 veces más rápida que la frecuencia de la señal y generar al menos 10 ciclos o más. Esta configuración recomendada sería la de Figura B.7 donde se observa la señal senoidal de 1Hz configurada con una rata de generación de 50 muestras/seg y 500 muestras interpoladas con un retenedor de orden cero.

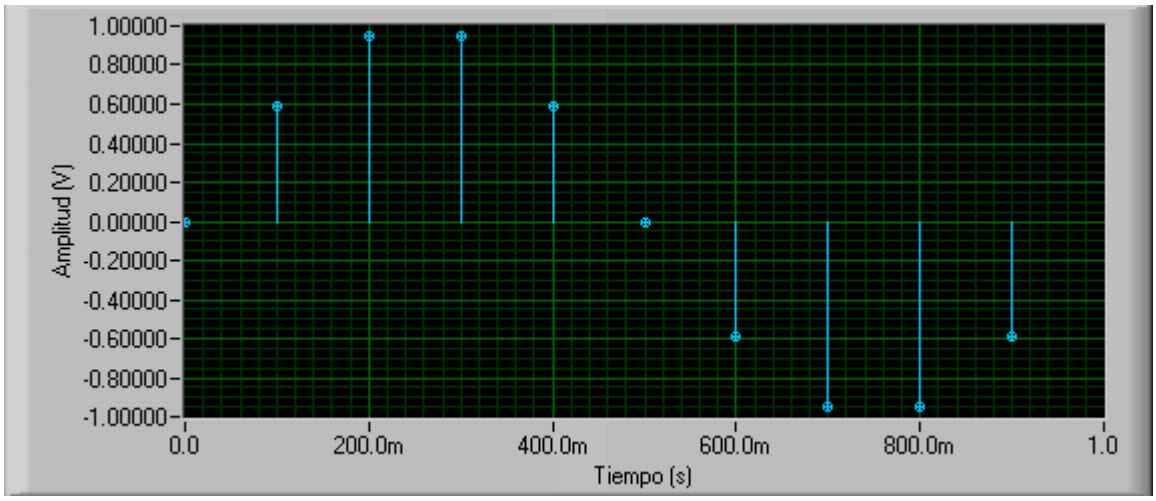


Figura B.1. Muestras senoidal de 1Hz. Rata de generación=10.Tamaño del *buffer*=10

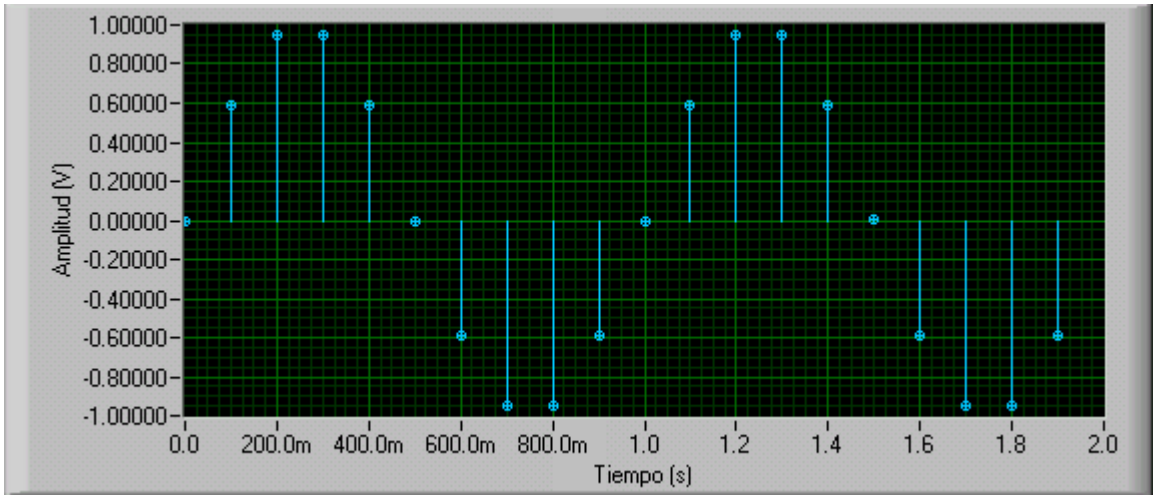


Figura B.2 Muestras senoidal de 1Hz. Rata de generación=10. Tamaño del *buffer*=20. La duración del *buffer* de salida en este caso es de 2s. Observese que la resolución vertical es la misma y la resolución horizontal se ha duplicado a 20 muestras.

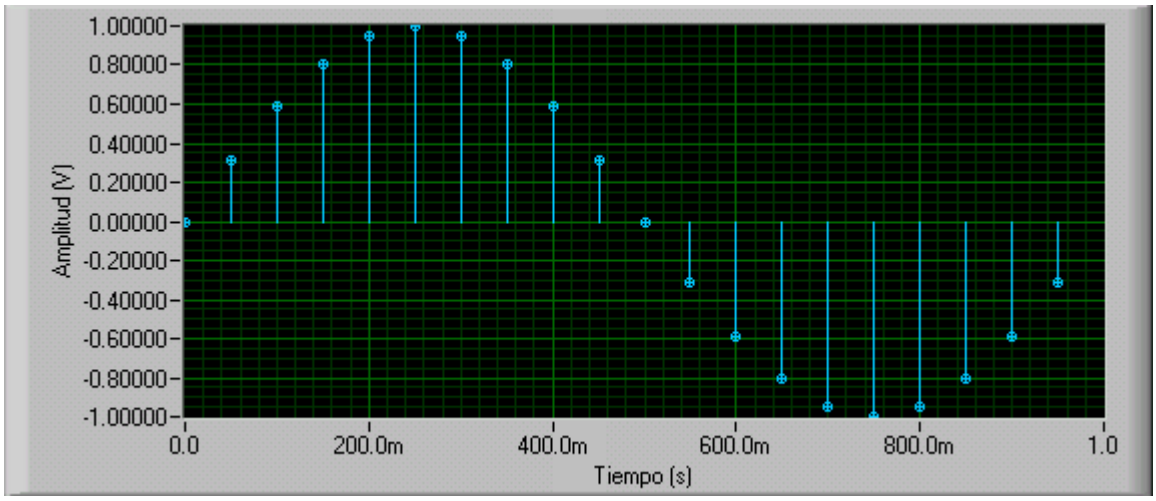


Figura B.3 Muestras senoidal 1Hz. Rata de generación=20. Tamaño del *buffer*=20. La duración del *buffer* es de 1s. La resolución vertical aumentó mientras que la resolución horizontal sigue siendo de 20 muestras.

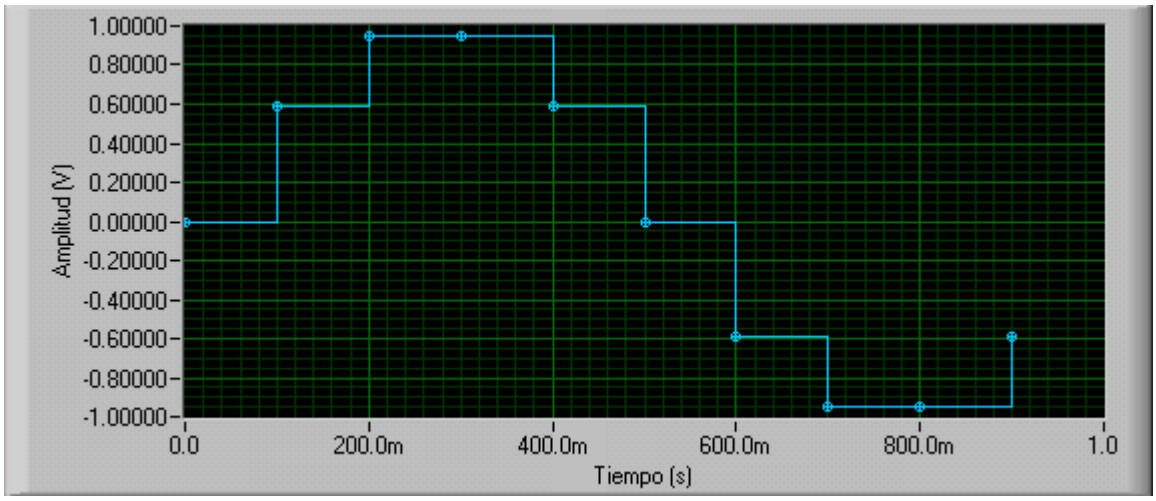


Figura B.4 Señal interpolada con retenedor de orden cero. Se observa que este tipo de reconstrucción es bastante abrupto aunque es suficiente para algunas aplicaciones y fines didácticos. Esta señal se puede mejorar luego de un filtrado paso bajo que elimine las componentes de alta frecuencia producidas por el rizado. En este caso la resolución vertical es de solo 2 niveles de voltaje.

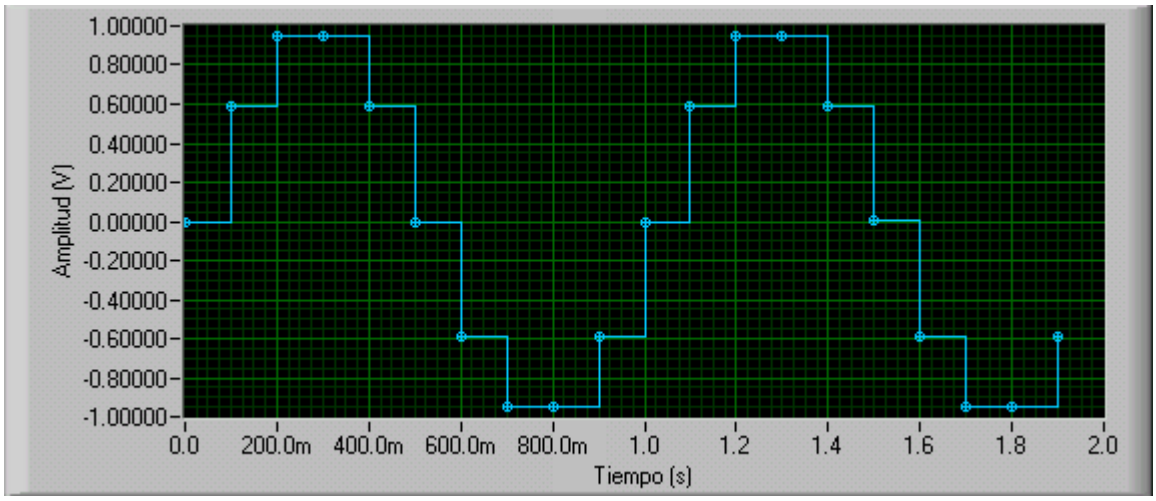


Figura B.5 Señal interpolada con retenedor de orden cero. Se observa que la resolución horizonta se duplicó con respecto a la figura anterior, sin embargo la resolución vertical se mantiene en 2 niveles de voltaje.

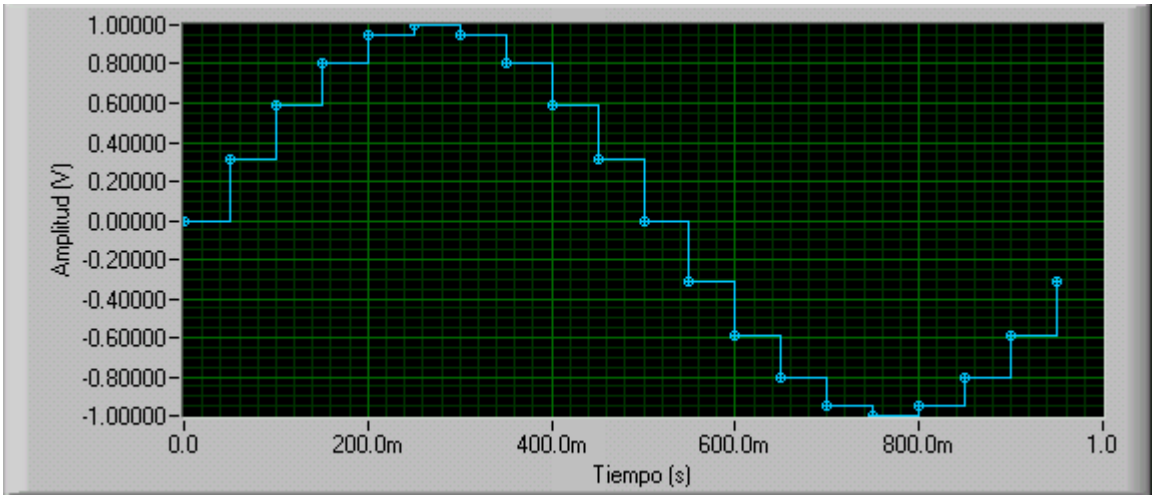


Figura B.6 Señal interpolada con retenedor de orden cero. Se observa que la resolución horizontal es la misma de la figura anterior pero en este caso la resolución vertical es de 5 niveles de voltaje lo cual me representa una mejor reconstrucción de la señal aunque se siguen notando cambios abruptos.

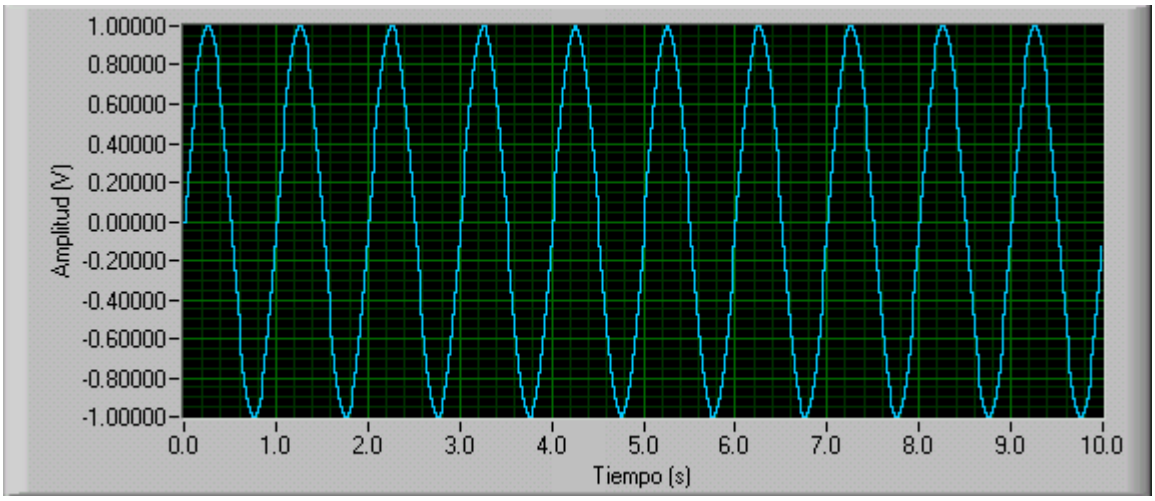


Figura B.7 Señal senoidal de 1Hz. Rata de generación=50, Tamaño del *buffer*=500. Interpolada con retenedor de orden cero. En esta figura se observa que la resolución vertical es tan grande que a pesar de utilizar un retenedor de orden cero para reconstruir la señal los niveles de voltaje son prácticamente impercibibles. La resolución horizontal es de 500 muestras para tener información de frecuencia confiable.

El ***Tipo de generación*** define como se generará el *buffer* de salida. Si se selecciona generación ***Continua*** el *buffer* se genera repetidamente. Si se selecciona generación ***Finita*** se genera únicamente un número entero de *buffer*s. Este control se observa en la Figura B.8.

El control para ***Almacenar en canal*** define por cual canal de salida se generarán señales analógicas. Si el usuario edita formas de onda en ambos canales la generación será simultánea. De otra forma la generación se hará por un solo canal. El control se observa en la Figura B.8.

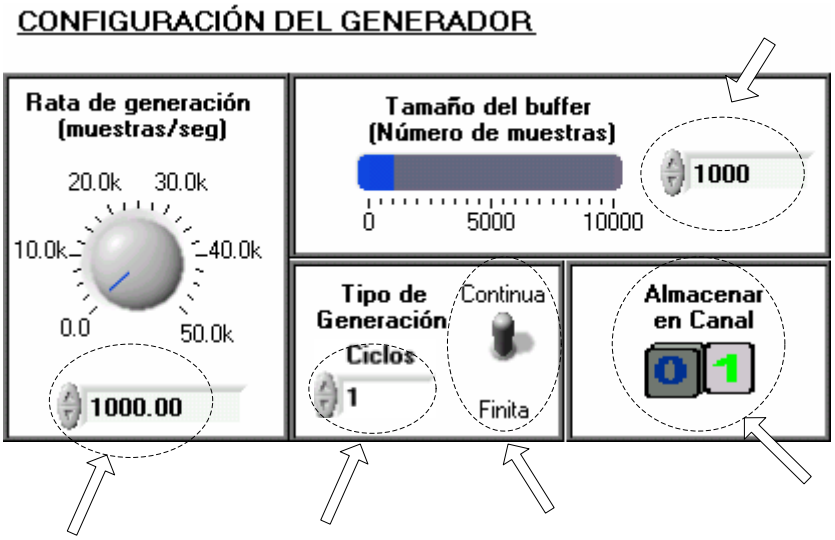


Figura B.8 Controles para configurar el generador.

2. SELECCIONAR LA OPCIÓN

Luego de configurado el generador se puede escoger entre las ocho opciones disponibles en el menú de selección ubicado en la parte superior izquierda. Las opciones son:

Función de librería

Al seleccionar esta opción se abrirá un panel (Ver Figura 2.8) en el cual se puede editar la forma de onda. Esta opción es la indicada cuando se desan generar formas de onda básicas como senoidales, triangulares, cuadradas y sierras. En la figura se observa una señal cuadrada editada en esta opción. Al terminar de editar su señal presione ***Aceptar*** para regresar al pane frontal del generador. En la Figura B.9 se observa una señal cuadrada editada con esta opción.

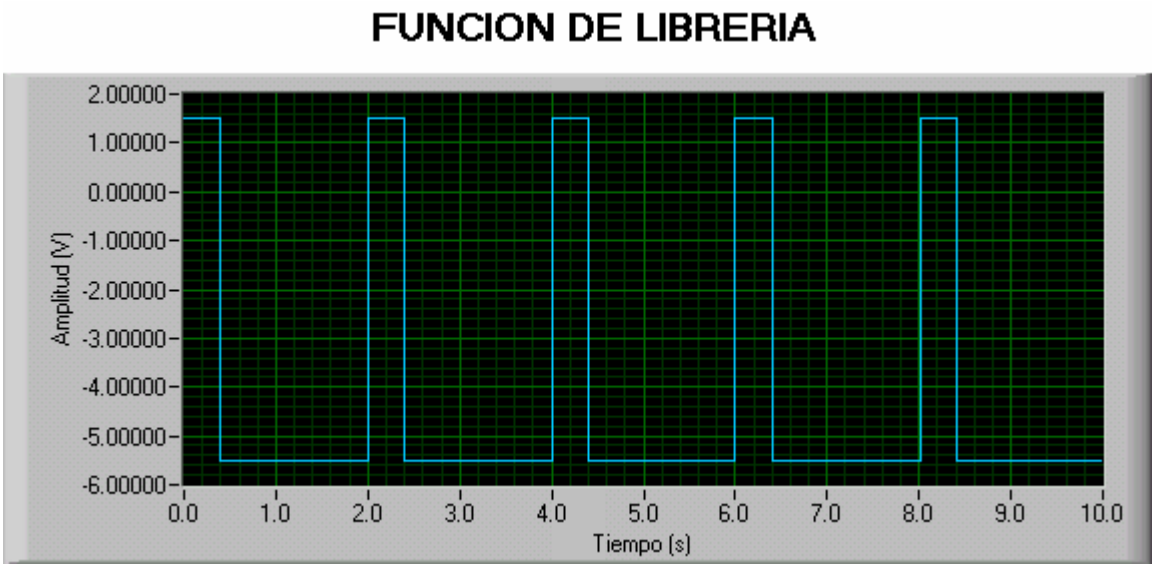


Figura B.9 Señal cuadradra editada en función de librería. Rata de generación=1000. Tamaño del *buffer*=10000. Amp=3.5V. Frec=0.5Hz. *Offset*= -2V. Ciclo útil=20%.

Función de Formula

Al seleccionar esta opción se abrirá un panel (Ver Figura 2.11) en el cual se puede editar la fórmula de la onda. Esta opción permite editar formas de onda no convencionales que se pueden realizar con operaciones aritméticas entre señales básicas como senoidales, cosenoidales o exponenciales. En la Tabla B.1 se describen algunas de las funciones integradas de LabVIEW las cuales se pueden utilizar para la edición de la fórmula. La fórmula se puede ingresar utilizando variables algebraicas a las cuales se les asigna el valor del control correspondiente. Las variables predeterminadas por LabVIEW son: “a” para amplitud, “f” para frecuencia, “w” para frecuencia angular (w=2*pi*f), “t” para el tiempo y “fs” para frecuencia de muestreo. En la Figura B.10 se observa la señal Abs(1+a*cos(w*t)+2*a*cos(2*w*t)) editada en esta opción. Al terminar de editar su señal presiones aceptar para regresar al pane frontal del generador.

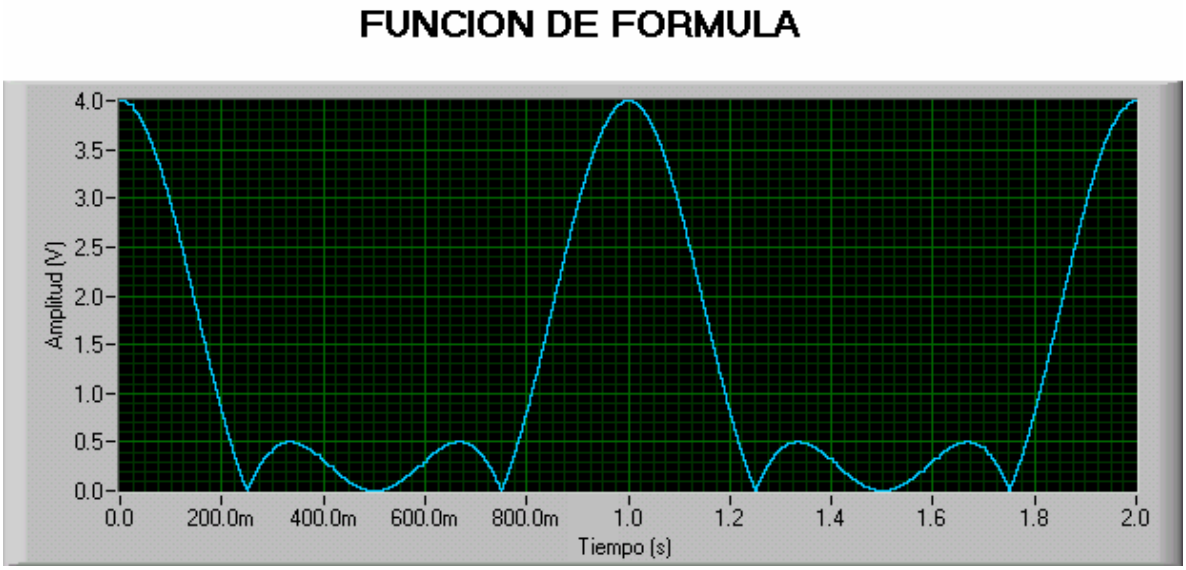


Figura B.10. Señal de fórmula. Amplitud=1V. Frecuencia =1Hz.
Función resultante: Abs(1+cos(2*pi*t)+2*cos(4*pi*t))

Tabla B.1. Algunas funciones matemáticas de LabVIEW

Función en LabVIEW	Descripción
sin(x),cos(x),tan(x),asin(x) acos(x),sinh(x),cosh(x), asinh(x),atanh(x),sinc(x)	Funciones trigonometricas, sus inversas, funciones hiperbólicas y funciones sinc
Abs(x),floor(x),exp(x)	Valor absoluto, parte entera, función exponencial.
Ln(x),log(x),log2(x)	Logaritmo natural, base 10 y base 2.
Pow(x,y),sqrt(x)	Computa x elevada a la y. Raíz cuadrada

Función de Mouse

Al seleccionar esta opción se abrirá un panel (Ver Figura 2.14) en el cual se puede dibujar una forma de onda con el mouse de 1000 muestras. Presione Aceptar para volver al panel principal. En esta opción el número de muestras es invariable con un valor de 1000. La señal dibujada representará un ciclo y por tanto para variar la frecuencia de salida es necesario variar la rata de generación. En la Figura B.12 observa un ejemplo de

un período de una señal de 1Hz, configurada en modo continuo con rata de generación de 1000.

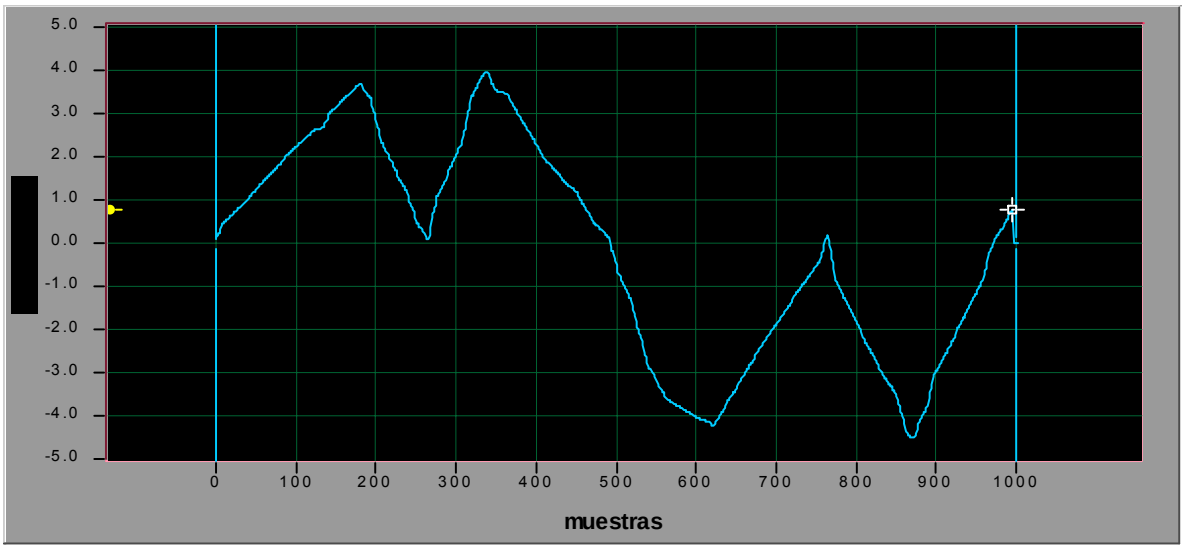


Figura B.12 Señal editada con el mouse

CH 0 + CH 1 = CH 0

Esta opción suma las señales del canal 0 y el canal 1 y el resultado lo guarda en el canal 0. Esta opción es útil para adicionar ruido de 60Hz o de alta frecuencia, sin perder el comportamiento en frecuencia de la señal original.

CH 0 - CH 1 = CH 1

Esta opción resta las señales del canal 0 y el canal 1 y el resultado lo guarda en el canal 1.

Guardar Señal

Guarda la señal del canal seleccionado en un archivo de texto. Al seleccionar la opción se despliega un cuadro de diálogo común de windows para seleccionar la ubicación donde se va a guardar la forma de onda. Escriba un nombre al archivo seguido de la extensión .txt y presione Guardar en la ventana de windows. Ejemplo: OndaSeno.TXT

Cargar señal

Carga una señal en el canal seleccionado desde un archivo de texto. Al seleccionar la opción se despliega un cuadro de diálogo común de windows para seleccionar la ubicación donde se encuentra guardado el archivo de texto con la información de la forma de onda. Seleccione el archivo y presione Abrir en la ventana de windows. En la Figura B.13 se observa la simulación de una arritmia cardíaca cargada desde un archivo de texto.

Advertencia: Si luego de seleccionar la opción 6 o 7 decide cancelar la operación. LabVIEW desplegará un mensaje de error notificando que la tarea fue cancelada. Presione *Continue* si desea seguir utilizando el generador normalmente o *Stop* si desea abortar el programa.

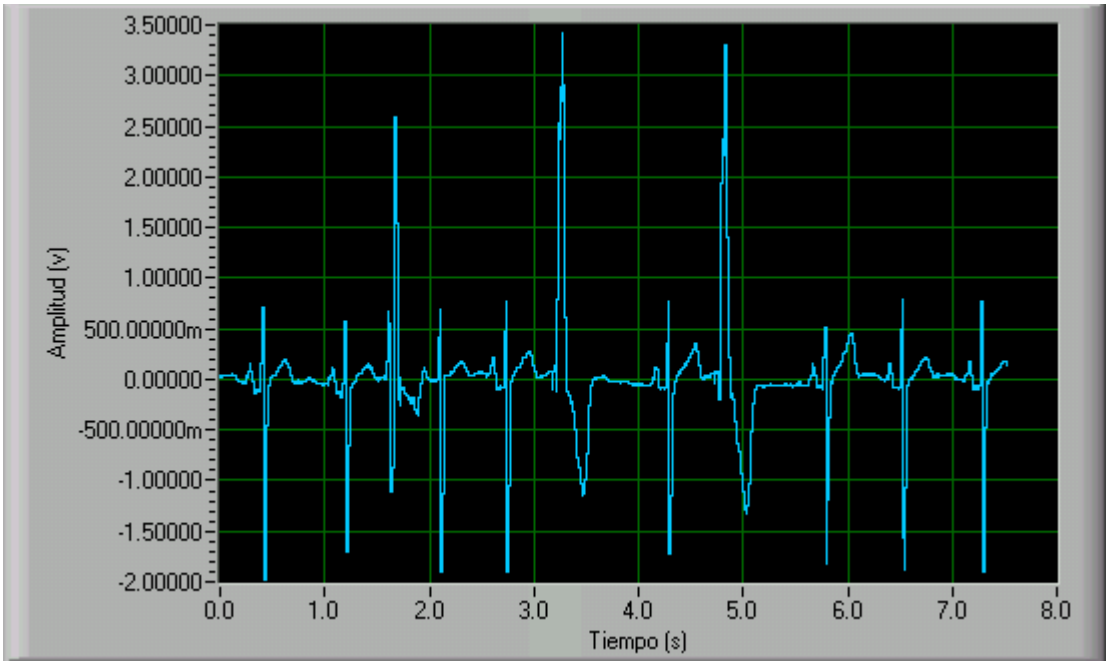


Figura B.13 Simulación de arritmia cardíaca cargada de base de datos. Rata de generación=125 muestras/s. Tamaño del *buffer*=941 muestras. (Modificación de simulación cortesía del codirector del proyecto).

Limpiar Canal

Borra la forma de onda almacenada en el canal seleccionado.

3. VERIFICAR LA FORMA DE ONDA.

Luego de editar la onda, se mostrará en el panel principal del generador una gráfica de lo que sería un *buffer* completo de salida. En la gráfica de las ondas de salida la señal del canal 0 es de color azul y la del canal 1 de color verde.

Revise también los indicadores de ***Estado de la generación*** ubicados en la parte inferior derecha del panel frontal del generador. Allí puede observar una frecuencia de salida estimada de la última señal cargada. Si ha cargado un archivo de texto hay un indicador con el número de muestras que tiene este archivo el cual debe ser el mismo que el tamaño del *buffer* para tener una frecuencia congruente. Estos indicadores se observan en la Figura B.14.

4. INICIE LA GENERACIÓN

Si esta satisfecho con la señal editada presione el control *INICIAR GENERACIÓN* y la generación dará inicio. Si desea hacer cambios a la señal o desea editar una señal nueva, presione y repita los pasos 1 a 3. También es posible cambiar el tipo de generación si se desea y en este caso excepcional no debe repetir los pasos 1 a 3 si no directamente iniciar la generación luego de hacer el cambio. Si desea información sobre el número de muestras *buffers* y muestras generadas durante la generación revise los indicadores ubicados en ***Estado de la generación*** los cuales se observan en la Figura B.14***.***

Nota: En tipo de generación continua, la generación se detendrá hasta que presione el botón DETENER GENERACIÓN. En tipo de generación finita, la generación se detendrá automáticamente al terminarse de generar todos los ciclos.

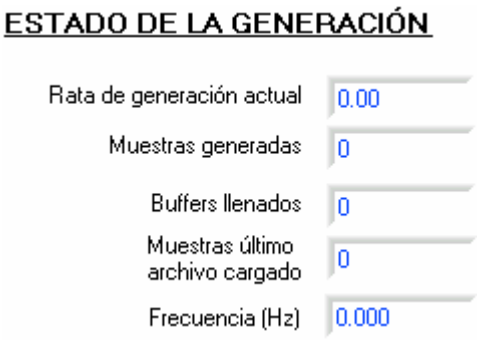


Figura B.14 Indicadores del Estado de generación

Si desea terminar totalmente la ejecución del programa, debe detener la generación y presionar el control *Salir*. Para obtener una ayuda rápida sin necesidad de salirse del programa presione el control *Ayuda.* (Véase Figura 2.3 para ubicar estos controles).

B.2 FILTRADO DIGITAL EN TIEMPO REAL Y SIMULACIÓN

Lo primero que debe hacer el usuario es verificar si la tarjeta de adquisición de datos está correctamente instalada y configurada. Esto en el caso que vaya a ejecutar la aplicación de tiempo real. El de simulación no la requiere. Luego debe revisar la presencia de los archivos que contienen los subprogramas de los que hace uso la aplicación. Estos se muestran con sus respectivos tamaños en disco en la Tabla B.2

Tabla B.2. Subvis necesarios para la ejecución de tiempo real.vi y simulación.vi

Subvi	Tamaño en disco
SubDisIIR.vi	405 kB
SubDisFIR.vi	204 kB
SubProcIIR.vi	55 kB
SubProcFIR.vi	57 kB
SubCtrlMuestreo.vi	35 kB
Opc-Bas.vi	30 kB
Guia del usuario.vi	15 kB

VarGlob1.vi	82 kB

Además si se va a correr la aplicación de simulación se deben agregar los *subvis* correspondientes al generador de simulación. Estos se muestran en la Tabla B.3.

Tabla B.2. Subvis necesarios para la ejecución de tiempo real.vi y simulación.vi

Subvi	**Tamaño en disco**
Generador.vi	48 kB
SubVI Cargar Archivo.vi	47 kB
SubVI formula.vi	79 kB
SubVI Librería.vi	75 kB

Tiempo real.vi ocupa 189 kB en disco duro y simulacion.vi ocupa 137 kB.

Toda la carpeta de filtros digitales suma 1.41 MB.

El panel frontal de esta aplicación se muestra en la Figura B.15 y está compuesto de cuatro secciones que son:

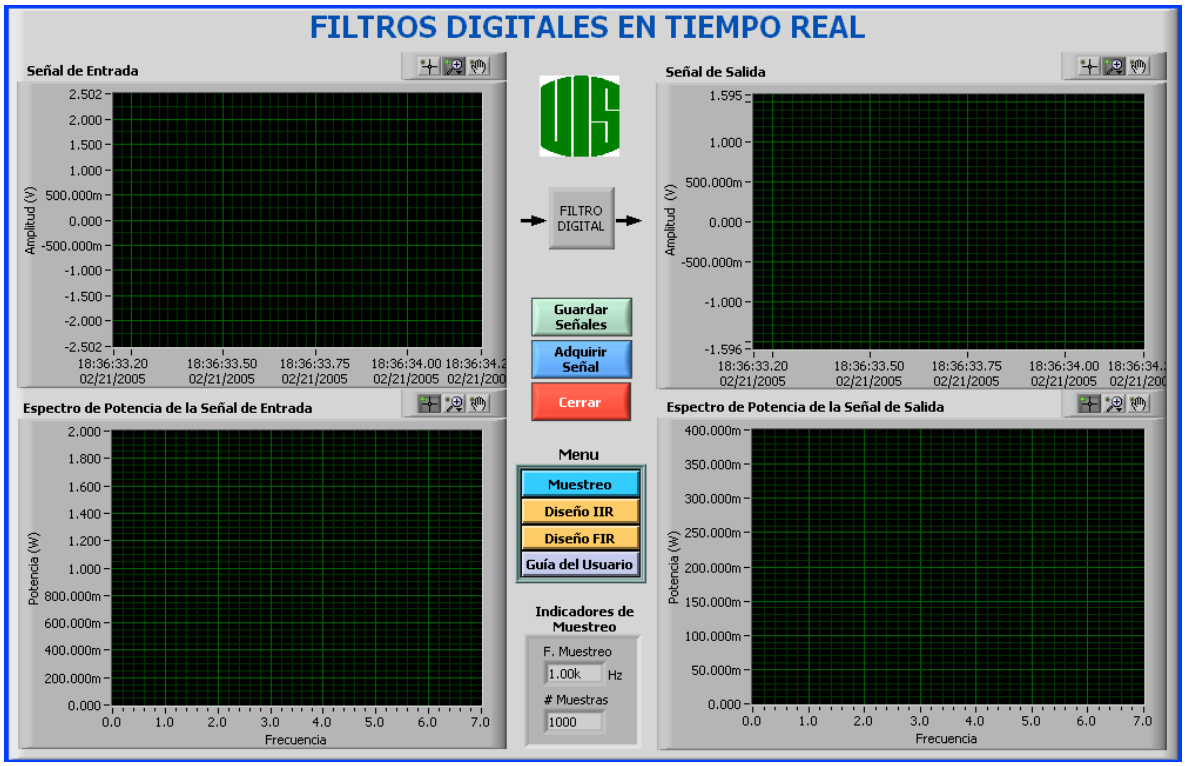


Figura B.15. Panel frontal de Tiempo real.vi.

- **Sección de Gráficas:**

  Esta sección se encuentra en la parte superior del panel frontal y contiene 4 gráficas que permiten la visualización de la señal de entrada, la señal de salida y sus respectivos espectros de potencia.

- **Sección de interruptores**

  Esta sección contiene tres interruptores:

  - **Adquirir señal:** Empieza o termina la adquisición.

  - **Guardar señales:** Guarda las señales de entrada y salida.

- **Cerrar:** Termina la ejecución del programa.

- **Menú**

  El menú permite ingresar a las aplicaciones de diseño y a la guía del usuario.

  - **Muestreo:** Ingresa a la aplicación donde se fijan los parámetros de muestreo.

  - **Diseño IIR:** Ingresa a la aplicación de diseño de filtros IIR.

  - **Diseño FIR:** Ingresa a la aplicación de diseño de filtros FIR.

  - **Guía del Usuario:** Contiene información sobre la operación del programa.

- **Indicadores de Muestreo**

Está compuesto por dos indicadores: frecuencia de muestreo y número de muestras. Estos indicadores tienen la función de recordar al usuario, la información de muestreo que se está utilizando.

ESTABLECIMIENTO DE LOS PARÁMETROS DE MUESTREO

Este *subvi* permite ingresar los parámetros de adquisición como: frecuencia de muestreo, número de muestras, tamaño del *buffer* y el rango límite de la señal de entrada. El usuario no tiene control sobre algunos parámetros como el dispositivo de adquisición de datos y el canal de adquisición. Estos permanecen fijos en el algoritmo y corresponden al dispositivo 1 y el canal 0.

El panel frontal de la presente aplicación se exhibe en la Figura B.16.

Figura B.16. Panel frontal de los parámetros de muestreo.

Este panel contiene cuatro controles que determinan la adquisición de la señal. Estos son: frecuencia de muestreo, número de muestras a leer, tamaño del *buffer* y límites de entrada.

- **Frecuencia de muestreo**

  La frecuencia de muestreo está dada en muestras por segundo y determina la velocidad a la cual el dispositivo de adquisición muestrea la señal analógica de entrada.

- **Muestras a leer**

  El número de muestras a leer determina la cantidad de muestras que la aplicación recupera del *buffer* para procesarlas.

- **Tamaño del *Buffer***

El tamaño del *buffer* determina la cantidad de memoria principal que se destina para el *buffer* de adquisición. Se recomienda que sea el doble del número de muestras a leer.

- **Límites de entrada**

Este control determina el rango de voltaje al que la TAD limita la señal de entrada, es decir que, si la señal sobrepasa estos límites, la tarjeta de adquisición satura la señal en los límites establecidos.

DISEÑO IIR

La rutina implementada para diseñar filtros IIR está organizada en dos secciones: coeficientes y especificaciones. La sección coeficientes permite diseñar a partir de los coeficientes de la ecuación en diferencias del filtro. La sección de especificaciones permite diseñar a partir de parámetros como: las frecuencias de corte, el orden, el tipo de filtro y la topología.

El usuario puede escoger cualquiera de las dos opciones (coeficientes o especificaciones), accionando el interruptor de barra traviesa que se muestra en el panel frontal de la Figura B.17.

Este panel está dividido en dos partes: La sección de diseño (izquierda) y la sección de Gráficas (derecha).

En la sección de diseño se introducen los parámetros de diseño basados en las especificaciones del filtro o los coeficientes de la ecuación de diferencias que caracteriza al mismo.

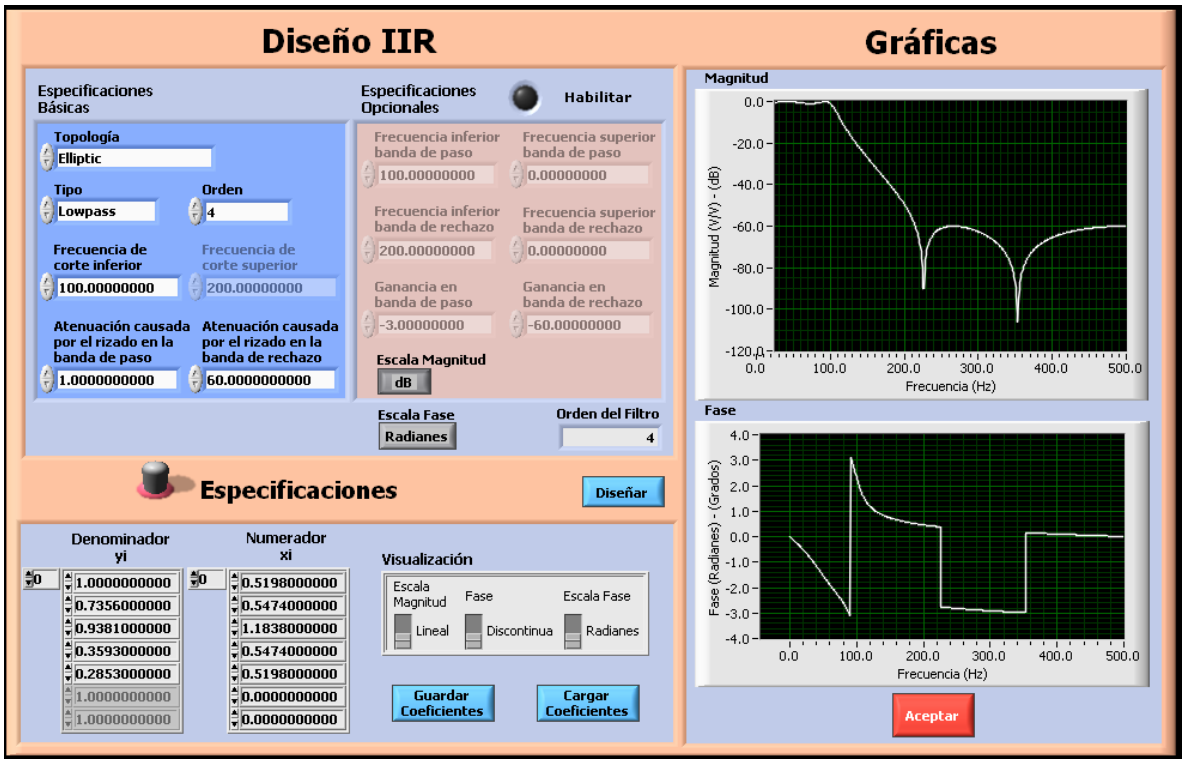


Figura B.17. Panel frontal del diseño IIR.

En la sección de gráficas se pueden observar las curvas de magnitud y fase del filtro diseñado.

A continuación se presenta la descripción de las opciones de diseño de filtros IIR que se manejan en esta rutina.

DISEÑO POR ESPECIFICACIONES

Las especificaciones desarrolladas en este proyecto para diseñar filtros IIR son dos: las básicas y las opcionales. Las especificaciones básicas son: la topología, el tipo de filtro (pasa bajas, pasa altas, pasa banda, rechaza banda), el orden, las frecuencias de corte y la atenuación en las bandas de paso y rechazo por causa del rizado. Ver Figura B.18.

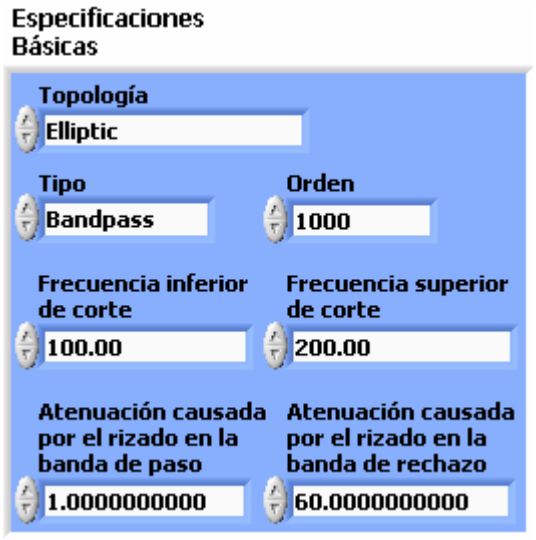


Figura B.18. Especificaciones básicas de diseño IIR. Tomada de LabVIEW.

- **Topología**

La topología del filtro es la función matemática que se utiliza para diseñar el filtro. Las más conocidas y que se encuentran disponibles en esta aplicación son: Butterworth, Chebyshev, Chebyshev inversa, Elíptica y Bessel.

- **Tipo**

El tipo de filtro puede ser pasa bajas, pasa altas, pasa banda o rechaza banda.

- **Orden**

  El orden del filtro se estableció como un valor entre 1 y 2500 para evitar el colapso del programa al utilizar valores superiores a este.

- **Frecuencia inferior de corte**

  Es la frecuencia a la cual se produce una ganancia de -3 dB.

- **Frecuencia superior de corte**

  También es una frecuencia a la cual la ganancia es de -3 dB pero sólo se utiliza cuando el filtro necesita dos frecuencias de corte como en el caso del filtro pasa banda y el rechaza banda.

- **Atenuación causada por el rizado en la banda de paso**

  Es el valor de atenuación producido por la adición de rizado en la banda de paso. Este valor obedece a la Ecuación 3.1

$$Atenuación = -20 * \log_{10}\left(\frac{Ao(f)}{Ai(f)}\right) \qquad \text{(B.4)}$$

  La topología Elíptica permite rizado en la banda de paso y la banda de rechazo, la de Chebyshev sólo permite rizado en la banda de paso y la de Chebyshev Inversa lo hace en la de rechazo. Las topologías Butterworth y Bessel no presentan rizado en ninguna de sus bandas

- **Atenuación causada por el rizado en la banda de paso**

  Tiene el mismo significado de la anterior, con la única diferencia de que el rizado se presenta en la banda de rechazo.

Las especificaciones opcionales se utilizan cuando no se conoce el orden que pueda cumplir con los requerimientos de ganancia y frecuencia del filtro. En este caso el programa calcula el orden a partir de los requerimientos mencionados. Las especificaciones opcionales se presentan en la Figura B.19.

Figura B.19. Especificaciones opcionales de diseño IIR.

- **Frecuencia inferior de la banda de paso**

Este valor de frecuencia controla el límite inferior (más a la izquierda) de la banda de paso.

- **Frecuencia superior de la banda de paso**

Este valor de frecuencia corresponde al límite superior (más a la derecha) de la banda de paso.

- **Frecuencia inferior de la banda de rechazo**

Este valor contiene la menor frecuencia en la banda de rechazo.

- **Frecuencia superior de la banda de rechazo**

Este valor controla el límite superior de la banda de rechazo.

- **Ganancia en la banda de paso**

Este valor contiene la ganancia a la cual se encuentran las frecuencias de la banda de paso.

- **Ganancia en la banda de rechazo**

Este valor contiene la ganancia a la cual se encuentran las frecuencias de la banda de rechazo.

- **Escala de Magnitud**

Este control permite visualizar la gráfica de magnitud en dB o en V/V. También cambia la interpretación de los valores de ganancias. Es decir que, los valores deben ser acordes a la escala.

- **Escala Fase**

Este control permite visualizar la gráfica de fase en radianes o en grados.

- **Orden del filtro**

Este indicador muestra el orden computado por las especificaciones opcionales.

DISEÑO POR COEFICIENTES

Este tipo de diseño consiste en introducir los coeficientes de los polinomios numerador y denominador de H(Z), en dos vectores columna. Los controles que se utilizan en esta aplicación, para ingresar los coeficientes, se muestran en la Figura B.20.

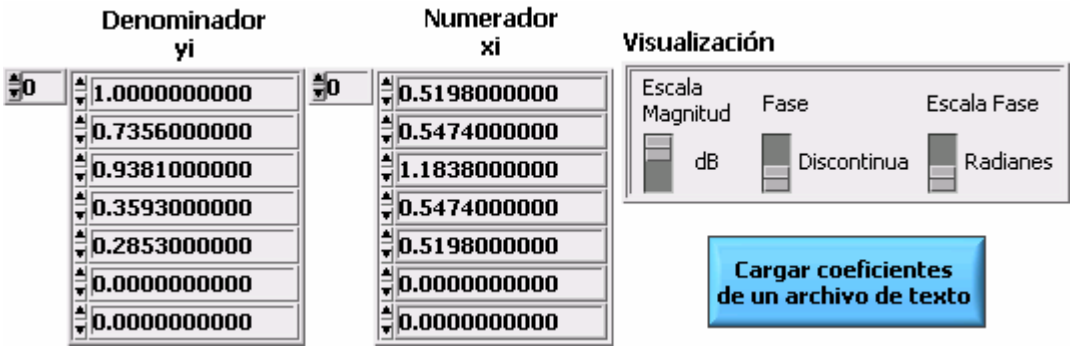


Figura B.20. Controles para introducir los coeficientes de H(Z).

Cada casilla del vector tiene un formato numérico de punto flotante con 10 decimales, con el fin de ofrecer al usuario, la posibilidad de tener una precisión alta en el valor de cada coeficiente.

También se tiene la posibilidad de cargar los dos vectores de un archivo de texto, accionando el control "Cargar coeficientes de un archivo de texto". En este archivo, los coeficientes deben ir en dos columnas separadas por una tabulación. La primera columna contiene los coeficientes del denominador de la función de transferencia del filtro y la segunda los coeficientes del numerador.

Los controles de "visualización" permiten cambiar la escala de la curva de magnitud, la escala de la fase y si la fase debe ser continua o discontinua.

DISEÑO FIR

La rutina implementada para diseñar filtros FIR está organizada en dos secciones: coeficientes y especificaciones. La sección coeficientes permite diseñar a partir de los coeficientes de la ecuación de diferencias del filtro. La sección de especificaciones

permite diseñar a partir de parámetros como: las frecuencias de corte, el orden, el tipo de filtro y la topología. El panel frontal se muestra en la Figura B.21.

Figura B.21. Panel frontal del diseño FIR.

Diseño por especificaciones

Las especificaciones se muestran en la Figura B.22.

- **Topología**

  La topología del filtro es el método que se utiliza para diseñar el filtro. Esta aplicación comprende dos métodos: ventaneado y rizado constante. Hay una topología llamada “FIR by specification” que implementa el método de rizado constante y permite hallar el orden del filtro a partir de los requerimientos de ganancia y frecuencia del filtro.

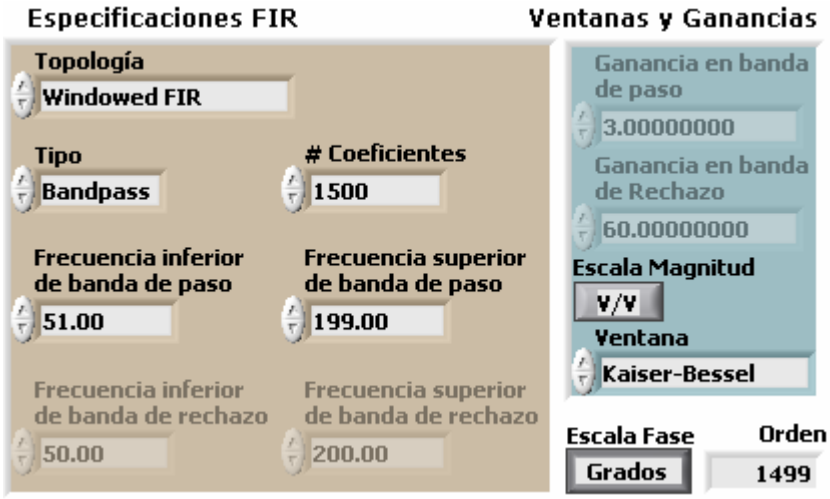


Figura B.22. Especificaciones básicas de diseño FIR. Tomada de LabVIEW

- **Número de coeficientes**

  El número de coeficientes es un valor entre 2 y 1500 y está relacionado con el orden del filtro mediante la expresión $(\#coeficientes = orden + 1)$. El número de coeficientes se limitó a 1500 para evitar el colapso del programa al utilizar valores superiores a este.

El diseño por coeficientes se hace de la misma manera que en el diseño IIR con la diferencia de que sólo se debe introducir el vector numerador.

Para ejecutar la adquisición y el procesamiento de la señal se debe presionar “adquirir” o en el caso de simulación “simular”. Para detener este proceso se debe presionar este mismo control.

ANEXO C: TEORÍA DE SISTEMAS DISCRETOS

C.1 CONCEPTOS BÁSICOS SOBRE SISTEMAS DISCRETOS

El tema principal de este proyecto son los FILTROS DIGITALES, los cuales extraen o modifican alguna característica de la señal de entrada; que se ve reflejada en su espectro de frecuencia, por ejemplo se habla del paso de una determinada banda del espectro de la señal de entrada. Por lo tanto, los filtros digitales se definen como sistemas discretos que modifican características en frecuencia de una señal digital.

Ante esta definición, es necesario comprender conceptos básicos de los sistemas discretos.

Las señales discretas en el tiempo surgen en la mayoría de los casos (Steiglitz, 1965; Oppenheim y Jonson, 1972) si el sistema involucra la operación de muestreo periódico de señales en tiempo continuo, como se explica a continuación.

Sea $x_C(t)$ una señal continua en el dominio del tiempo como se observa en la Figura C.1., entonces se obtiene una secuencia de muestras:

$x[n] = x_C(nT), \quad -\infty < n < \infty$, para n entero

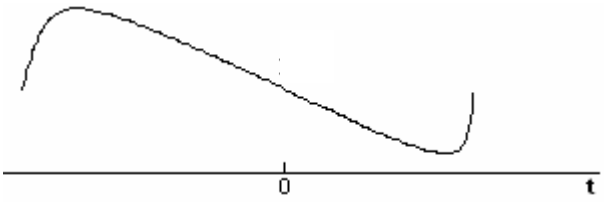


Figura C.1. Señal analógica $x_C(t)$

Siendo T el periodo de muestreo y fs = 1/T la frecuencia de muestreo en unidades de muestras por segundo.

La secuencia de muestras son $x_C(0)$, $x_C(T)$, $x_C(2T)$ …… $x_C(nT)$ como se observa en la Figura C.2.

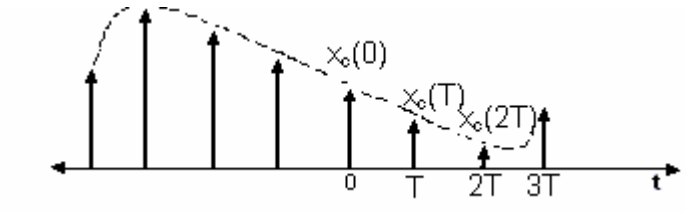


Figura C.2. Señal $x_C(nT)$

El proceso de muestro se puede entender mejor en el siguiente ejemplo

Ejemplo Muestreo de una señal continua en el tiempo

Obtener la secuencia de muestras de una señal sinusoidal de 100Hz, de amplitud 5V que ha sido muestreado con una frecuencia de 1kHz, sabiendo que el desfase en la adquisición es de 30º y el tiempo total de muestreo ha sido de 20 ms. No se consideran los efectos de cuantificación de la señal.

T = 1/fs = 1/1000 = 1ms
El periodo de la señal $x_C(t)$ = 1/100 = 10 ms
30º = (30º * 10 ms)/ 360º = 5/6 = 0.833333 ms
$x_C(t)$ = 0.5 Sen(2*pi*100*t+pi/6)) entonces x[n] = $x_C(nT)$ = $x_C(0)$, $x_C(T)$, $x_C(2T)$,..... $x_C(nT)$
x[n] = {0.25, 0.45, 0.49, 0.33, 0.5,-0.25, -0.45, -0.49, -0.33,-0.5, 0.25, 0.45, 0.49, 0.33, 0.5, -0.25, -0.45, -0.49, -0.33, -0.5}

SOLUCIÓN EN MATLAB

```
» A=0:20;T=1/1000;
» Y=0.5*sin((2*pi*100*(A*T)+pi/6));
» stem(A*T,Y)
```

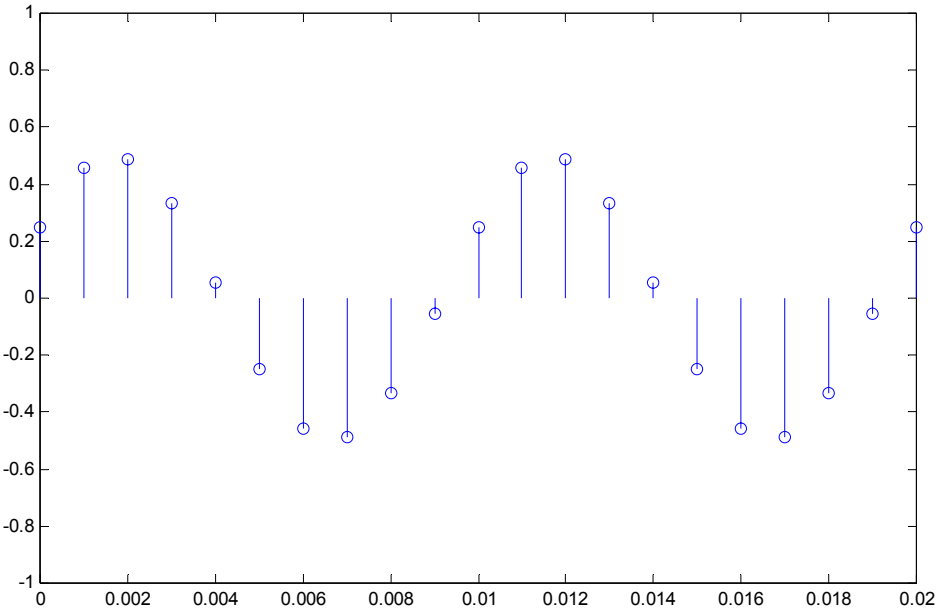


Figura C.3. Señal senoidal muestreada

Habiendo entendido ya lo que es una secuencia, la cual se asocio al muestreo, se puede interpretar que un sistema discreto es un algoritmo que permite convertir una secuencia discreta en otra y se puede expresar como se ilustra en la Figura C.3.

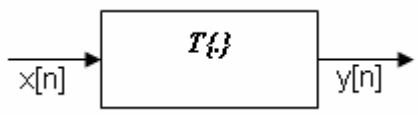


Figura C.4. Sistema discreto

Donde T{.} es el operador de transformación que convierte una secuencia de entrada x[n] en una secuencia de salida y[n].

Como ejemplo de sistema discreto se podría considerar cualquier programa de computador que transforme una tabla de valores en otra, por ejemplo, a obtención de cuadrados, la suma de los k primeros números, la media de una serie de muestras, etc.

SISTEMAS CAUSALES

Un sistema es causal si el valor de la secuencia de salida en el índice n = n_O; depende solo de los valores de la secuencia de entrada para n ≤ n_O, es decir el sistema solo depende de valores presentes y pasados de la señal de entrada.

$$y[n] = x[n] + x[n-2] \qquad \text{Sistema causal}$$

$$y[n] = x[n+1] + x[n]$$

En la Ecuación anterior el sistema es no causal ya que la entrada x[n+1] se esta anticipando a la salida.

SISTEMAS ESTABLES

Un sistema es estable si para una secuencia de entrada acotada la salida es también acotada.

Definición:

Sea Kx un valor positivo finito tal que

$$| x[n] | \leq Kx < \infty \quad \text{Para todo n}$$

Entones el sistema es estable si se cumple que

$$| y[n] | \leq Ky < \infty \qquad \text{Para todo n}$$

De estas ecuaciones se observa que la salida para un sistema estable siempre es acotada cuando la entrada es también acotada.

LINEALIDAD

Un sistema se considera lineal si cumple con el concepto de superposición o propiedad aditiva y la propiedad de homogeneidad o escalado tal como se ilustra en la Figura C.5, sea Y1[n] y Y2[n] las respuestas de un sistema para cuando las entradas son X1[n] y X2[n] respectivamente. Entonces para un sistema lineal se cumple que:

$$T\{X1[n] + X2[n]\} = T\{X1[n]\} + T\{X2[n]\} = Y1[n] + Y2[n]$$

$$T\{aX[n]\} = aT\{x[n]\} = aY[n]$$

$$T\{aX1[n] + bX2[n]\} = aT\{X1[n]\} + bT\{X2[n]\}$$

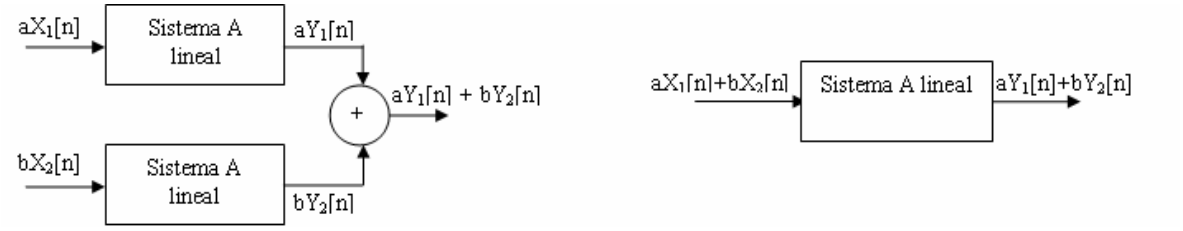


Figura C.5. Comportamiento de un sistema lineal

INVARIANZA EN EL TIEMPO

Un sistema es invariante en el tempo si un desplazamiento n_0 de la secuencia de entrada produce un desplazamiento n_0 en la secuencia de salida. Sea y[n] la respuesta de un sistema cuando la entrada es x[n], entonces:

$$T\{x[n-n_O]\} = y[n-n_O]$$

SISTEMAS LINEALES E INVARIANTES EN EL TIEMPO (LTI)

Un sistema es lineal e invariante con el tiempo cuando cumple con las condiciones de linealidad e invarianza, esta clase de sistemas es de particular importancia y se caracteriza por su respuesta al impulso.

$$Y[n] = \sum_{k=-\infty}^{\infty} X[k]h[n-k]$$ Suma de convolución

Siendo h[n] la respuesta del sistema cuando la entrada es un impulso δ [n].

Los sistemas que se tratan en este proyecto están orientados a sistemas LTI (sistemas lineales e invariantes con el tiempo).

MUESTREO DE SEÑALES EN TIEMPO CONTINUO

Las señales en tiempo discreto pueden aparecer de muchas formas, sin embargo la forma más común de que aparezcan es mediante el muestreo de señales en tiempo continuo.

MUESTREO PERIÓDICO

La forma típica de obtener una representación de una señal en tiempo discreto de una señal en tiempo continuo es mediante el muestreo periódico de la señal en tiempo continuo.

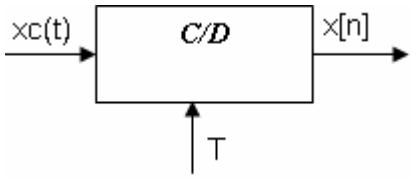


Figura C.6. Conversor ideal de tiempo continuo a tiempo discreto

En la práctica el proceso de muestreo se realiza mediante un conversor analógico-digital como el que se observa en la Figura C.6.

En general el muestro es una operación no invertible, es decir dada una señal en tiempo discreto x[n], no es posible reconstruir exactamente la señal analógica de la cual provino, existen muchas señales en tiempo continuo que pueden producir la misma secuencia de

muestras, sin embargo para reducir esta ambigüedad se limita la entrada de señales al sistema de muestreo (señales de banda limitada).

A continuación se detalla el proceso de muestreo, como se indica en la Figura C.7.

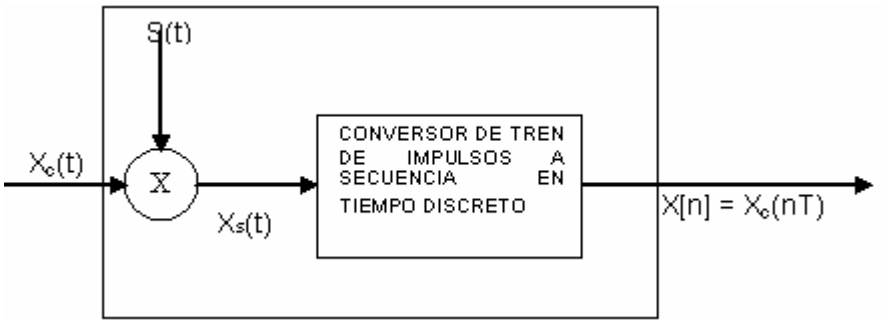


Figura C.7. Conversor análogo digital

Para representar matemáticamente el proceso de muestreo se separa en dos partes tal como indica la Figura C.7 la primera parte es un modulador con un tren de impulsos y la segunda es un conversor de tren de impulsos a secuencia en tiempo discreto.

Sea la señal $x_C(t)$ como lo indica la Figura C.8

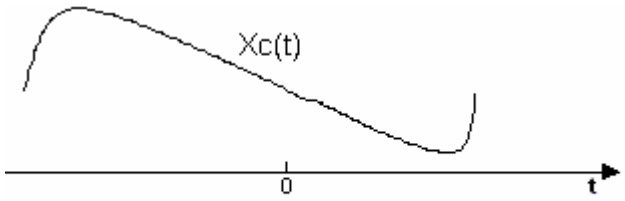


Figura C.8 Señal $x_C(t)$ en tiempo continúo

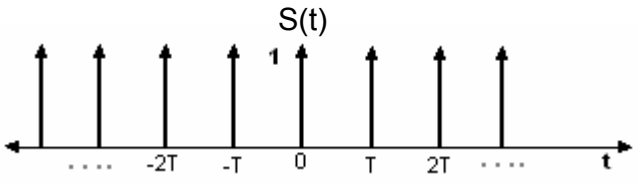


Figura C.9 Tren de impulsos en el dominio del tiempo de periodo T ≤ la mitad del periodo de la señal $x_C(t)$

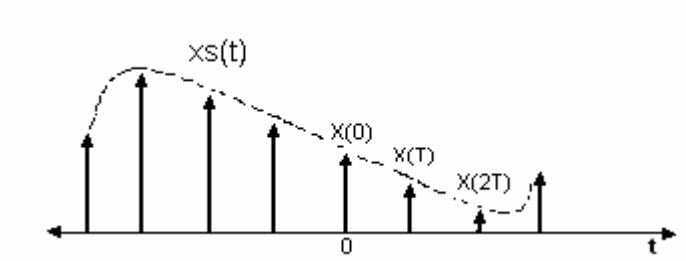


Figura C.10. Señal $x_C(t)$ modulada con tren de impulsos en el dominio del tiempo

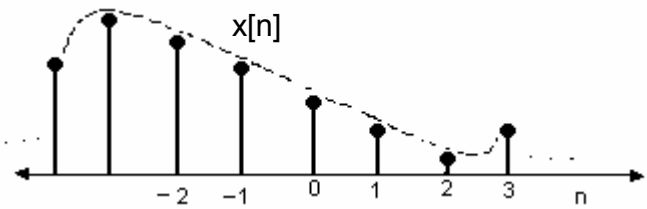


Figura C.11. Salida del conversor de tren de impulsos a secuencia en tiempo discreto

De las Figuras C.8 a la C.11 podemos observar que la diferencia fundamental entre Xs(t) y x[n] es un escalamiento en el eje de las x, también nótese que x[n] no proporciona ninguna información acerca de la frecuencia de muestreo (fs).

REPRESENTACIÓN DEL MUESTREO EN EL DOMINIO DE LA FRECUENCIA

Se quiere expresar x[n] en términos de la señal de entrada $x_C(t)$ en el dominio de la frecuencia, de un conversor C/D ideal como el de la Figura C.6.

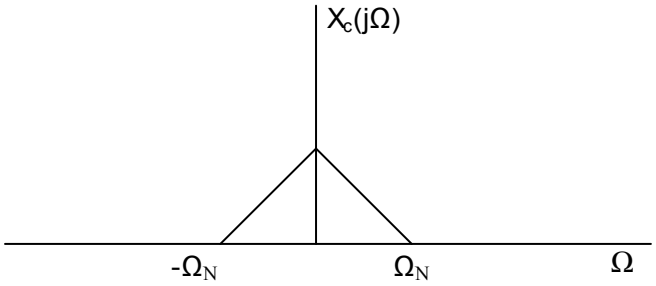


FIGURA C.12. Transformada continua de Fourier de $x_C(t)$

$S(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$ Siendo δ (t) la función impulso unidad.

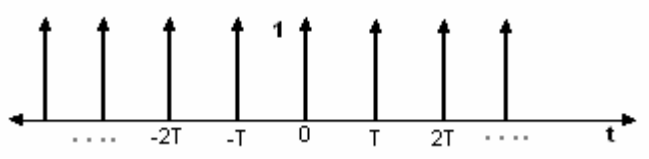


FIGURA C.13. Tren de impulsos. T es el periodo de muestreo

$$S(j\Omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\Omega - k\Omega_s)$$

Transformada continua de Fourier del tren de impulso de la Figura C.13. Donde Ωs es $\frac{2\pi}{T}$

$$X_s(t) = X_c(t)S(t) = X_c(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) = \sum_{n=-\infty}^{\infty} X_c(nT)\delta(t - nT)$$

$$X_s(j\Omega) = \frac{1}{2\pi} X_c(j\Omega) * S(j\Omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c(j(\Omega - k\Omega_s)) = \sum_{n=-\infty}^{\infty} X_c(nT)e^{-j\Omega nT}$$

Transformada continua de Fourier de $X_s(t)$.

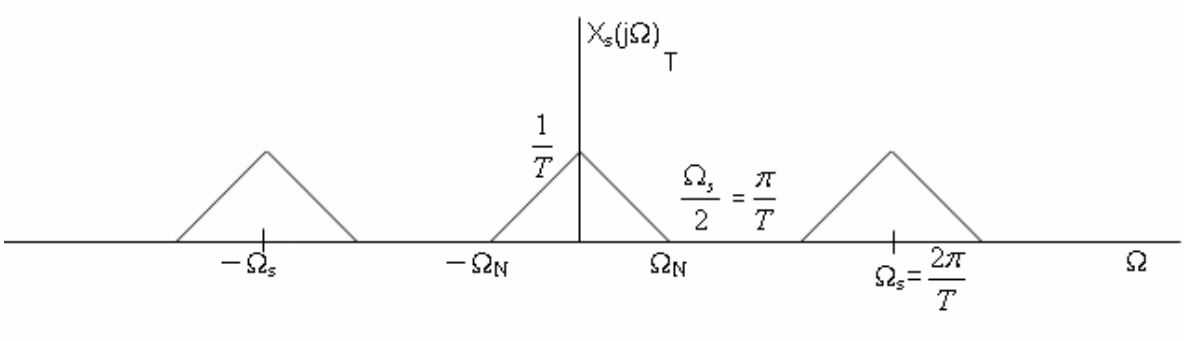


FIGURA C.14. Transformada de Fourier continua de $x_s(t)$

De la Figura C.14 puede verse claramente que en frecuencia el proceso de modular la señal $x_c(t)$ con el tren de impulsos es repetir la señal $x_c(j\Omega)$ cada Ω_s desde -∞ al ∞, además también se observa que para que la señal original $x_c(j\Omega)$ no se altere es necesario que:

$\Omega_s > 2\Omega_N$, Teorema de Nyquist

Si $\mathbb{W}_s$ es ≤ $\mathbb{W}_N$ se presenta el fenómeno de aliasing o solapamiento como se indica en la Figura C.15.

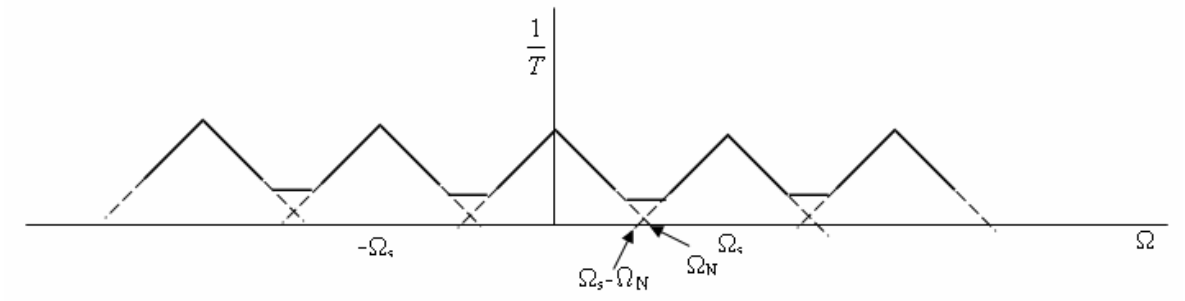


FIGURA C.15. Fenómeno de Aliasing

El fenómeno de aliasing se muestra en la Figura C.15. Ocurre cuando la frecuencia de muestreo $f_s \leq f_N$, donde f_N es la frecuencia máxima de la señal continua a muestrear y se conoce como frecuencia de Nyquist.

Como x[n] = $x_c(nT)$ y la transformada de Fourier de x[n] es

$$X(e^{jw}) = \sum_{n=-\infty}^{\infty} X[n]e^{-jwn}$$

Entonces se deduce que

$$X_s(j\Omega) = X(e^{jw})_{w=\Omega T} = X(e^{j\Omega T})$$

Aplicando se tiene que:

$$X(e^{j\Omega T}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_c(j(\Omega - k\Omega_s)) \text{ Y } X(e^{jw}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_c(j(\frac{w}{T} - \frac{2.\pi.k}{T}))$$

En esta ecuación se muestra la transformada discreta de Fourier de x[n] en términos de $x_C(j\Omega)$.

La transformada de Fourier de la señal $x_S(t)$ y x[n], es simplemente una versión escalada en frecuencia definido por w = ΩT, siendo w la frecuencia discreta y Ω la frecuencia continua.

RECONSTRUCCIÓN DE SEÑALES DE BANDA LIMITADA A PARTIR DE SUS MUESTRAS.

Al reconstruir una señal a partir de sus muestras lo primero a tener en cuenta es que al sistema de muestreo debe entrar una señal de banda limitada, esto significa que se debe conocer el valor de la frecuencia máxima (f_N) de la señal a muestrear, esta limitante depende del sistema en general como por ejemplo de la velocidad de muestreo de la tarjeta de adquisición de datos. Teniendo en cuenta estos aspectos se debe diseñar y construir un filtro antialiasing, que garantice un valor máximo de frecuencia, de esta forma podremos definir un valor de frecuencia de muestreo, que satisfaga todas las señales de entrada. Bajo este supuesto podemos reconstruir una señal a partir de sus muestras con

un filtro pasa bajas ideal de frecuencia de corte $\frac{\pi}{T}$ como se ilustra en la Figura C.17., el proceso en general esta caracterizado como se indica en la Figura C.16.

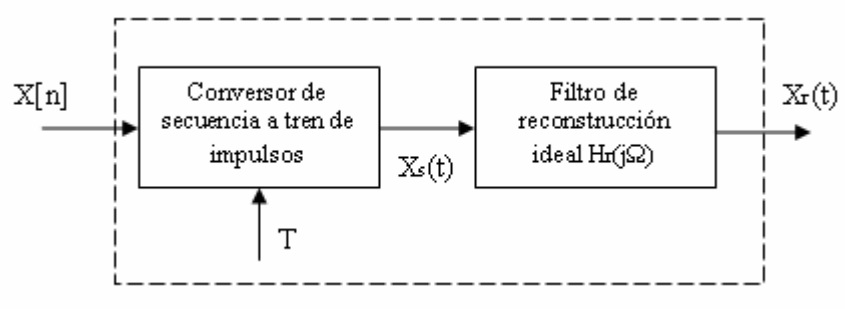


FIGURA C.16. Sistema de reconstrucción ideal

El sistema de reconstrucción ideal lo forma en general un conversor de secuencia a tren de impulsos, seguido de un filtro de reconstrucción ideal.

$x_S(t) = \sum_{n=-\infty}^{\infty} \frac{X[n]}{\delta(t - nT)}$, ahora si este tren de impulsos es la entrada a un filtro de reconstrucción ideal pasa bajos entonces la salida del filtro será:

$$X_r(t) = \sum X[n] h_r(t - nT)$$

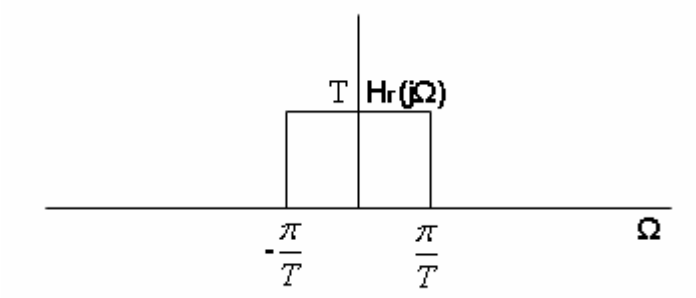


FIGURA C.17. Transformada de Fourier continua del filtro de reconstrucción ideal

La transformada inversa de Fourier de Hr(jΩ) es una Sinc de amplitud T como se muestra en la ecuación:

$$h_r(t) = \frac{sen(\frac{\pi t}{T})}{\frac{\pi t}{T}}$$

Entonces se tiene que la salda del filtro es

$$X_r(t) = \sum X[n] \frac{sen\left[\frac{\pi(t-nT)}{T}\right]}{\frac{\pi(t-nT)}{T}}$$

Ahora calculando la transformada de Fourier de la anterior ecuación se tiene:

$$X_r(j\Omega) = \sum_{n=-\infty}^{\infty} x[n] H_r(j\Omega) e^{-j\Omega Tn} = H_r(j\Omega) X\left(e^{j\Omega T}\right)$$

PROCESADO EN TIEMPO DISCRETO DE SEÑALES EN TIEMPO CONTINUO

Una aplicación importante de los sistemas en tiempo discreto es el procesado de señales en tiempo continuo tal como lo indica la Figura C.18.

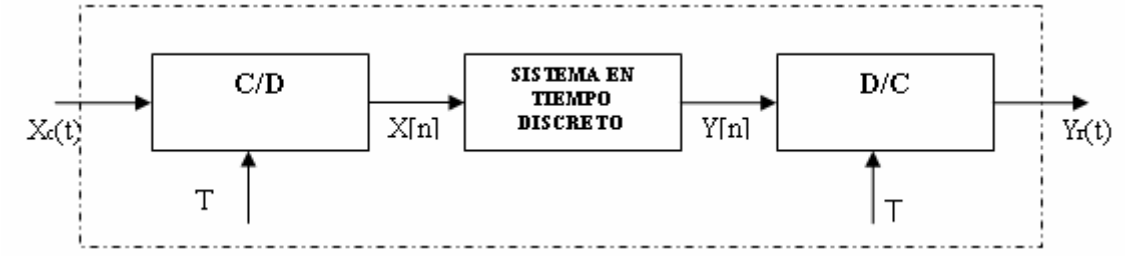


FIGURA C.18. Sistema ideal de procesado en tiempo discreto de señales en tiempo continuo.

La salida del conversor C/D es x[n] = $x_C(nT)$ y su transformada de Fourier es

$$X(e^{jw}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_c(j(\frac{w}{T} - \frac{2.\pi.k}{T}))$$

La salida del conversor D/C es $Y_r(t) = \sum_{n=-\infty}^{\infty} Y[n] \frac{sen\left[\frac{\pi(t-nT)}{T}\right]}{\frac{\pi(t-nT)}{T}}$

Entonces aplicando se tiene

$$Y_r(j\Omega) = H_r(j\Omega) Y(e^{j\Omega T})$$

= T $Y(e^{jWT})$, $|\Omega| < \frac{\pi}{T}$ y cero en el resto

Si el sistema de la Figura C.18 es un sistema LTI entonces

$$Y(e^{j\Omega T}) = H(e^{jw}) X(e^{jw})$$

Remplazando se tiene

$$Y_r(j\Omega) = H_r(j\Omega) H(e^{jw}) X(e^{jw})$$

Si se observa la Figura C.17, del filtro ideal de reconstrucción $H_r(j\Omega)$ se tiene

$$Y_r(j\Omega) = H(e^{jw}) X_c(j\Omega).$$ para |w| < $\frac{\pi}{T}$ y cero en le resto.

TRANSFORMADA Z

La transformada Z constituye la herramienta más utilizada en el análisis y diseño de sistemas en tiempo discreto, siendo la operación análoga a la transformada de Laplace para el estudio de sistemas en tiempo continuo.

Para un sistema discreto se puede tener una caracterización a través de una ecuación de diferencias. Para determinar la respuesta del sistema a una entrada previamente aplicada, es necesario resolver la ecuación de diferencias. Con el método de la transformada Z, la solución de la ecuación de diferencias se convierte en un problema de naturaleza algebraica. Por consiguiente, el empleo de la transformada Z permite hacer un análisis de las características de un determinado filtro definido en tiempo discreto, lo que se traduce en que éste método facilita el análisis y diseño de sistemas a través de funciones de transferencia en el dominio Z, en general un sistema representado en el dominio de Z se caracteriza por

$$Y(z) = X(z)H(z)$$

La variable compleja z es una transformación no lineal de la variable w de Fourier, con el propósito de que la función de transferencia del sistema discreto obtenida de esta transformación sea racional y abarque un mayor número de secuencias que no abarca la transformada de Fourier

La transformada Z bilateral de la secuencia x[n], se define:

$$X(z) = \sum_{n=-\infty}^{\infty} X[n] z^{-n}$$

Donde z^{-n} es un operador que transforma una secuencia en una función X(z), siendo z una variable compleja.

Transformada Z unilateral de una secuencia x[n], se define:

$$X(z) = \sum_{n=0}^{\infty} X[n] z^{-n}$$

La variable compleja Z se define

$$Z = re^{jw}$$

Donde r es la magnitud y w el ángulo. Como la variable Z es una función de variable compleja, es conveniente representarla gráficamente en el plano complejo Z.

PLANO Z

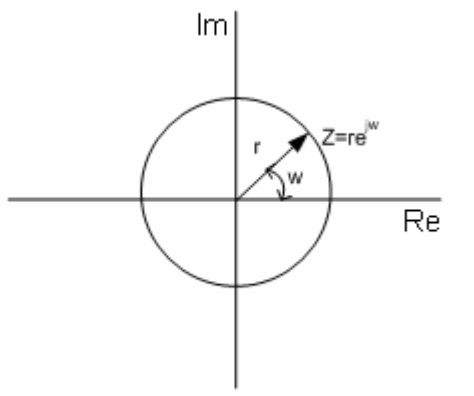


FIGURA C.19. Plano Z

Si se compara la transformada Z con la transformada de Fourier de una secuencia son iguales para $Z = e^{jw}$ entonces evaluar la transformada Z en la circunferencia de radio uno e es la transformada de Fourier.

La transformada de Fourier no converge para todas las secuencias, esa es la primera motivación para desarrollar la transformada Z, la cual converge para un número mayor de secuencias. La transformada Z tampoco converge para todas las secuencias ni para todos los valores de Z.

El conjunto de valores de Z para los cuales la transformada converge se denomina Región de convergencia.

El criterio de convergencia de la transformada Z es:

$$\sum_{n=-\infty}^{\infty} / X[n] // z /^{-n} < \infty$$

Como lo indica la ecuación la transformada Z converge cuando la sumatoria es absolutamente sumable.

La transformada Z tiene toda su utilidad cuando se puede expresar mediante una función cerrada de Z. Entre las transformadas Z más importantes se destacan aquellas que se pueden expresar de forma racional, es decir,

$$X(z) = \frac{P(z)}{Q(z)}$$

Los valores de Z para los cuales la ecuación es cero e infinito se denominan ceros y polos de X(z), respectivamente.

Ejemplo Transformada Z de una secuencia exponencial derecha

Sea la señal $x[n] = a^n u[n]$. La transformada Z es

$$X(z) = \sum_{n=-\infty}^{\infty} X[n] z^{-n} = \sum_{n=-\infty}^{\infty} a^n u[n] z^{-n} = \sum_{n=0}^{\infty} \left(az^{-1}\right)^n$$

Teniendo en cuenta el criterio de convergencia tenemos que:

$$\sum_{n=0}^{\infty} | \left(az^{-1}\right) |^n < \infty$$

Por lo tanto la transformada Z será el conjunto de valores que cumplan que |z|>|a| y su transformada es,

$$X(z) = \sum_{n=0}^{\infty} \left(az^{-1}\right)^n = \frac{1}{1-az^{-1}} = \frac{z}{z-a}, \qquad |z|>|a|$$

Ahora tenemos una función continua y una región de convergencia, con un cero en z=0 y un polo en z=a.

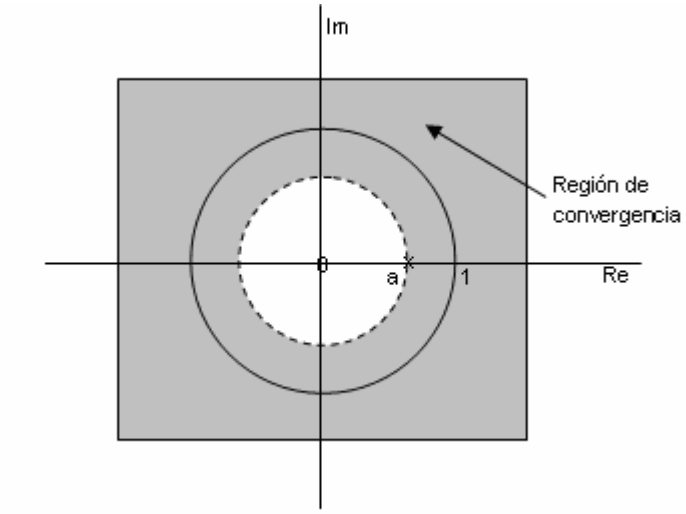


FIGURA C.20. Región de convergencia del Ejemplo 1.2.

De la Figura C.20 se observa que el valor de (a) determina la región de convergencia, por ejemplo si (a) es menor que 1, entonces la región de convergencia de la transformada z incluye la circunferencia unidad o lo que es lo mismo la transformada de Fourier converge, por el contrario si (a) es mayor que 1 la región de convergencia no incluye la transformada de Fourier.

Ejemplo Transformada Z de una exponencial Izquierda

Sea $x[n] = -a^n u[-n-1]$, entonces la transformada Z es:

$$X(z) = \sum_{n=-\infty}^{\infty} a^n u[-n-1] z^{-n} = -\sum_{n=-\infty}^{-1} a^n z^{-n} = -\sum_{n=1}^{\infty} a^{-n} z^n = 1 - \sum_{n=0}^{\infty} \left(a^{-1} z\right)^n$$

Del criterio de convergencia tenemos que |z|<|a|.

$$X(z) = \frac{1}{1 - az^{-1}} = \frac{z}{z - a} \quad , \qquad |z|<|a|$$

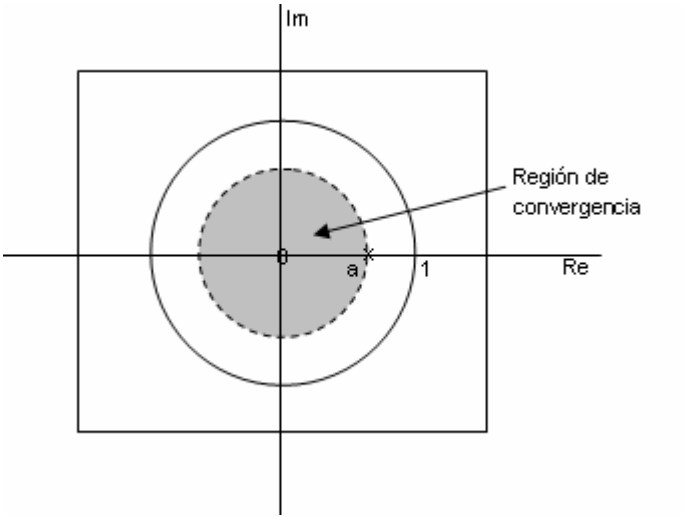


FIGURA C.21. Transformada Z del ejemplo

De los Ejemplos se observa que dos secuencias pueden tener la misma transformada Z excepto por su región de convergencia.

PROPIEDADES DE LA REGIÓN DE CONVERGENCIA

- La región de convergencia puede ser un anillo o un disco centrado en el origen.

- La transformada de Fourier de x[n] converge si y solo si la transformada Z incluye en su región de convergencia la circunferencia unidad.

- La región de convergencia no contiene ningún polo.

- Si x[n] es una secuencia de duración finita, entonces la región de convergencia es todo el plano Z excepto tal ves en Z=0 o en Z=∞

- Si x[n] es una secuencia limitada por la izquierda, entonces la región de convergencia se extiende desde el polo finito más exterior hacia fuera hasta el infinito pudiendo incluir el valor de Z=∞.

- Si x[n] es una secuencia limitada por la derecha, entonces la región de convergencia se extiende desde el polo finito mas interior hacia adentro hasta el cero pudiendo incluir el valor de Z= 0.

- Si x[n] es una secuencia bilateral, entonces la región de convergencia es un anillo.

- La región de convergencia debe ser continua.

- Del diagrama de polos y ceros y la región de convergencia se obtienen resultados interesantes como:

- para que un sistema sea estable, entonces h[n] debe ser absolutamente sumable, es decir debe tener transformada de Fourier, lo que significa que la región de convergencia de la transformada Z debe incluir la circunferencia unidad.

- Para que un sistema sea causal entonces h[n] debe ser limitada por la izquierda, lo que significa que la región de convergencia se extiende desde el polo mas exterior hasta z = ∞

LA TRANSFORMADA Z INVERSA

Se plantea el problema inverso, obtener la secuencia x[n] a partir de su transformada X(z). Para el caso de tener X(z) en forma de función racional, se puede calcular de forma sencilla. La fórmula analítica se define como:

$$X[n] = \frac{1}{2j\pi}\oint X(z)z^{n-1}dz$$

Método de inspección

El método de inspección consiste en estar familiarizado con la transformada Z de ciertas secuencias de tal forma que al ver la transformada Z identificamos la secuencia de la cual provino, este método es válido y se ayuda de las tablas de transformaciones, ya que muchas secuencias son bastante comunes como por ejemplo las exponenciales.

Descomposición en fracciones simples

Existen casos donde por simple inspección o por ayuda de las tablas, no es posible encontrar la transformada inversa, sin embargo si se pueden expresar los términos de forma simple se pueden encontrar más fácilmente las secuencias correspondientes a cada término, este es el caso de las funciones racionales, donde por medio del método matemático de descomposición en fracciones parciales es posible descomponer los términos racionales en una suma de fracciones simples.

Ejemplo Transformada inversa usando fracciones parciales

Se define un sistema discreto por la siguiente ecuación de diferencias:

y[n] = 3x[n] – 5/6 x[n-1] + 7/12 y[n-1] -1/12 y[n-2]

Calcular la transformada Z del sistema y la respuesta del sistema cuando la entrada es un escalón.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{3-\frac{5}{6}Z^{-1}}{1-\frac{7}{12}Z^{-1}+\frac{1}{12}Z^{-2}} = \frac{3-\frac{5}{6}Z^{-1}}{\left(1-\frac{1}{4}Z^{-1}\right)\left(1-\frac{1}{3}Z^{-1}\right)}$$

Se observa que la transformada Z de la función de transferencia tiene 2 polos ubicados en z = 1/4 y z = 1/3, respectivamente.

La transformada Z de la función escalón es.

$$\frac{1}{1-Z^{-1}}$$

Entonces la respuesta del sistema para cuando la entrada es un escalón es

$$Y(z) = \frac{1}{1-Z^{-1}} * \frac{3-\frac{5}{6}Z^{-1}}{\left(1-\frac{1}{4}Z^{-1}\right)\left(1-\frac{1}{3}Z^{-1}\right)}$$

La descomposición en fracciones simples resulta

$$Y(z) = -\frac{1/3}{\left(1-1/4Z^{-1}\right)} - \frac{1}{\left(1-1/3Z^{-1}\right)} + \frac{13/3}{1-Z^{-1}}$$

La transformada inversa es por simple inspección

$$-\frac{1}{3}\left(\frac{1}{4}\right)^n - \left(\frac{1}{3}\right)^n + \frac{13}{3} \quad u[n]$$

Resolución en Matlab

» u=ones(1,20);% vector escalón de 20 puntos

» x=0:19;

» y=filter([3,-5/6],[1,-7/12,1/12],u);% calculo de la salida cuando la entrad es un escalón

» stem(x,y); grafica de 20 puntos de la salida

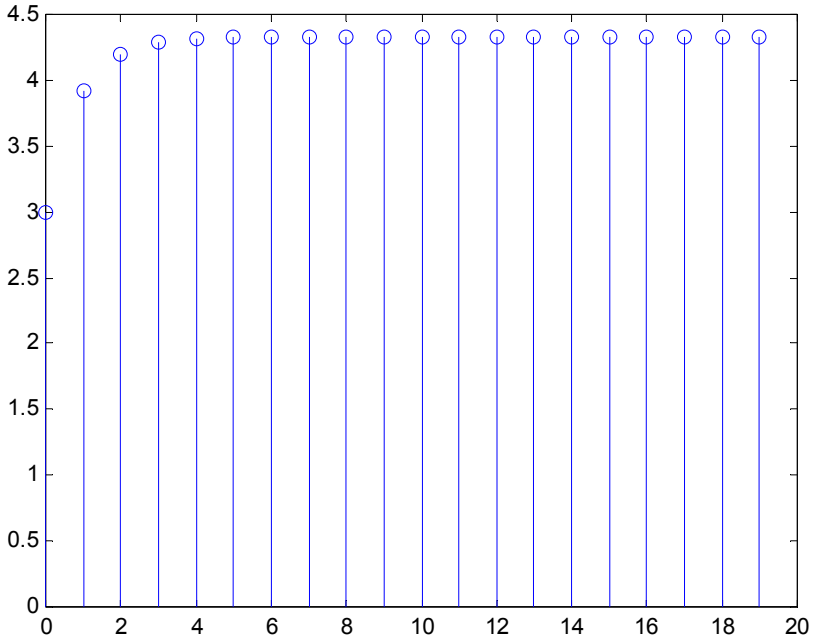


Figura C.22 Respuesta del Ejemplo 1.4

PROPIEDADES DE LA TRANSFORMADA Z

- LINEALIDAD

  $aX_1[n]+bX_2[n]\overset{z}{\leftrightarrow}aX_1(z)+bX_2(z)$, la región de convergencia es $R_{X1}\cap R_{X2}$

- DESPLAZAMIENTO EN EL TIEMPO

  $X[n-n_0]\overset{z}{\leftrightarrow}z^{-n_0}X(z)$, la región de convergencia es R_X (excepto por la posible adición o supresión de z=0 o z=∞.

- MULTIPLICACIÓN POR UNA SECUENCIA EXPONENCIAL

$z_o^n X[n] \overset{z}{\leftrightarrow} X\left(\frac{z}{z_o}\right)$, la región de convergencia es $|z_0|R_X$ y z_0 es re^{jwo} una exponencial compleja.

- DIFERENCIADOR DE X(z).

$nX[n] \overset{z}{\leftrightarrow} -z\frac{dX(z)}{dz}$, la región de convergencia es R_X

- CONJUGACIÓN DE UNA SECUENCIA COMPLEJA

$X^*[n] \overset{z}{\leftrightarrow} X^*(z^*)$, la región de convergencia es R_X

- CONVOLUCIÓN DE SECUENCIAS.

$X_1[n] * X_2[n] \overset{z}{\leftrightarrow} X_1[z]X_2[z]$, la región de convergencia contiene a $R_{X1} \cap R_{X2}$

ECUACIONES EN DIFERENCIAS CON COEFICIENTES CONSTANTES

Una subclase importante de los sistemas lineales e invariantes en el tiempo corresponde a los sistemas que se pueden representar mediante ecuaciones de diferencias con coeficientes constantes, de la forma

$$\sum_{k=0}^{N} a_k Y[n-k] = \sum_{m=0}^{M} b_m X[n-m]$$

Para los sistemas caracterizados por la anterior ecuación la salida se puede calcular directamente o recursivamente, si el sistema es causal o no causal respectivamente, además si el sistema es causal también será lineal e invariante en el tiempo.

De lo anterior se entiende que la ecuación de diferencias, la respuesta al impulso y la función de transferencia son todas caracterizaciones equivalentes de la relación entrada-salida de un sistema discreto, lineal e invariante en el tiempo, cuando se trabaja en un sistema digital por ejemplo un computador, es necesario transformar estas expresiones en un algoritmo aplicable a la tecnología que se tenga, para el caso de los computadores las ecuaciones de diferencias son de gran utilidad ya que se pueden formar con operaciones básicas como la suma, la multiplicación por una constante y el retardo

Ejemplo: Cálculo de la ecuación de diferencias a partir de la función de transferencia en Z

Se tiene la siguiente función de transferencia.

$$H(z) = \frac{b_o + b_1 z^{-1}}{1 - az^{-1}} \quad , \quad |z| > a$$

Su respuesta al impulso es

$$h[n] = b_o a^n u[n] + b_1 a^{n-1} u[n-1]$$

Como la respuesta al impulso es de duración infinita no es posible hacerlo mediante convolución discreta, sin embargo si usamos ecuaciones en diferencias tenemos.

$$Y[n] - aY[n-1] = b_o X[n] + b_1 X[n-1],$$

Ahora se rescribe la ecuación para obtener un algoritmo para el cálculo recursivo de la salida.

$$Y[n] = aY[n-1] + b_o X[n] + b_1 X[n-1]$$

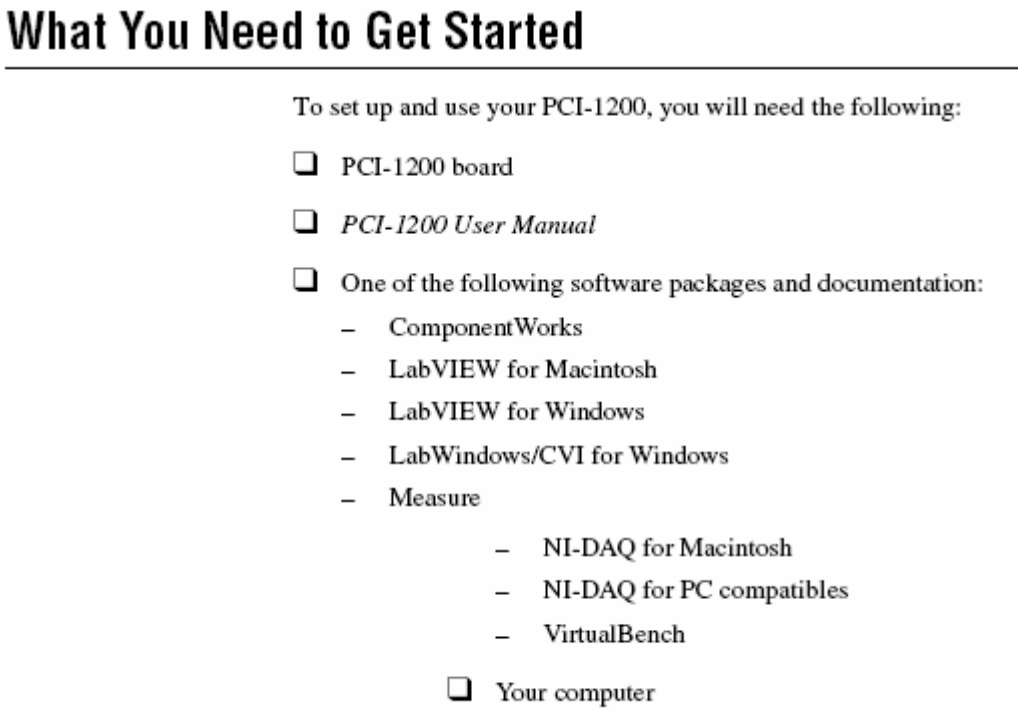
Para este algoritmo si se suponen condiciones iniciales de reposo el sistema será causal lineal e invariante en el tiempo. Este sistema ya se puede realizar en un sistema de computador con operaciones lógicas como la suma y espacio en memoria como el retardo.

La realización concreta de la estructura depende de la tecnología que se vaya a utilizar.

ANEXO D: ESPECIFICACIONES DE LA TARJETA DE ADQUISICIÓN DE DATOS PCI 1200 DE NATIONAL INSTRUMENTS

En este anexo se muestran los requerimientos necesarios para su funcionamiento, la configuración de pines y las especificaciones técnicas de entrada y salida análoga de la tarjeta de adquisición de datos.

REQUERIMIENTOS



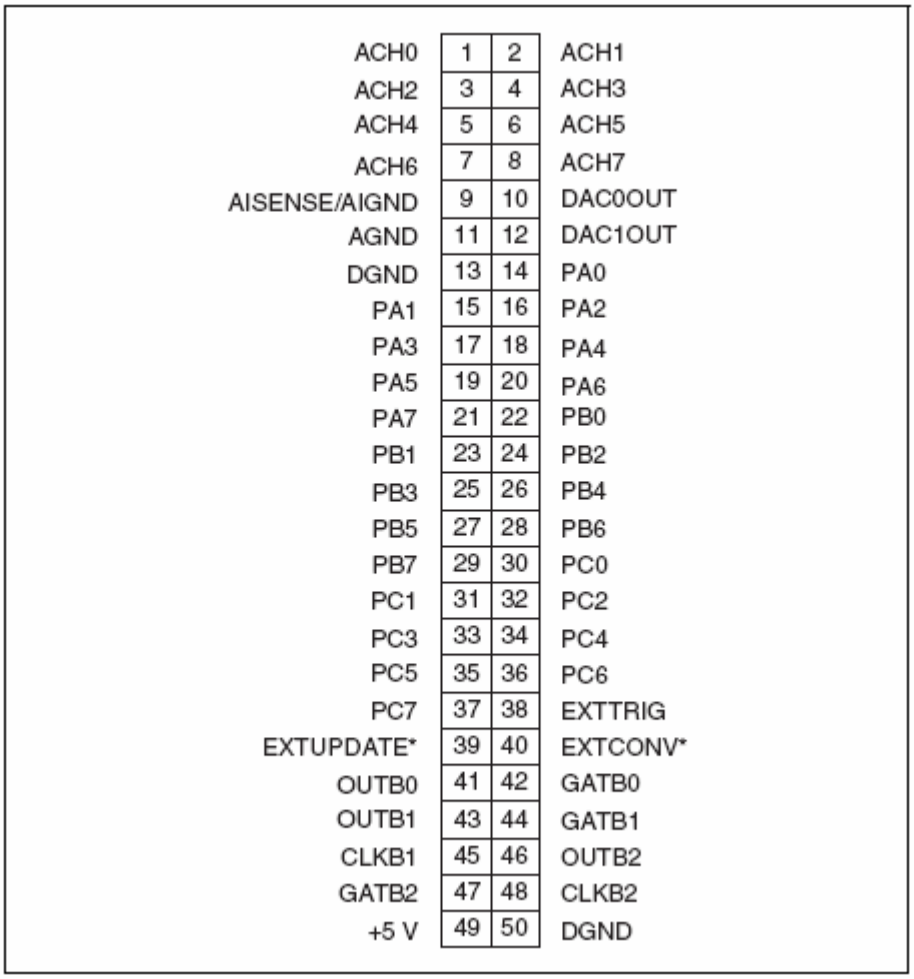
What You Need to Get Started

To set up and use your PCI-1200, you will need the following:

- ❑ PCI-1200 board
- ❑ *PCI-1200 User Manual*
- ❑ One of the following software packages and documentation:
  - ComponentWorks
  - LabVIEW for Macintosh
  - LabVIEW for Windows
  - LabWindows/CVI for Windows
  - Measure
    - NI-DAQ for Macintosh
    - NI-DAQ for PC compatibles
    - VirtualBench
- ❑ Your computer

Figura D.1. Requerimientos de la TAD PCI 1200. Tomada de *PCI 1200 User Manual.*

CONFIGURACIÓN DE PINES



ACH0	1	2	ACH1
ACH2	3	4	ACH3
ACH4	5	6	ACH5
ACH6	7	8	ACH7
AISENSE/AIGND	9	10	DAC0OUT
AGND	11	12	DAC1OUT
DGND	13	14	PA0
PA1	15	16	PA2
PA3	17	18	PA4
PA5	19	20	PA6
PA7	21	22	PB0
PB1	23	24	PB2
PB3	25	26	PB4
PB5	27	28	PB6
PB7	29	30	PC0
PC1	31	32	PC2
PC3	33	34	PC4
PC5	35	36	PC6
PC7	37	38	EXTTRIG
EXTUPDATE*	39	40	EXTCONV*
OUTB0	41	42	GATB0
OUTB1	43	44	GATB1
CLKB1	45	46	OUTB2
GATB2	47	48	CLKB2
+5 V	49	50	DGND

Figura D.2. Configuración de pines. Tomada de *PCI 1200 User Manual.*

De la Figura D.2 se describen en la Tabla D.1 los pines 1, 9, 10, 11 y 12 los cuales fueron los pines utilizados en este proyecto para la generación analógica y la adquisición de señales en tiempo real.

Tabla D.1. Descripción de los terminales de la TAD utilizados en el proyecto.

Número de pin	Nombre	Referencia	Descripción
1	ACH0	AGND	Canal cero de entrada analógica
9	AIGND	AGND	Tierra de entrada análoga. Conectada a AGND en modo RSE(Modo de entrada simple referenciada, por sus siglas en inglés). Este modo de entrada fue el utilizado en el proyecto
10	DAC0OUT	AGND	Canal cero de salida analógica
11	AGND	-	Tierra análoga. Referencia para señales de salida voltaje analógicas.
12	DAC1OUT	AGND	Canal uno de salida analógica

ESPECIFICACIONES DE ENTRADA Y SALIDA ANÁLOGA

Specifications

This appendix lists the PCI-1200 specifications. These specifications are typical at 25° C unless otherwise stated.

Analog Input

Input Characteristics

Number of channels ..... Eight single-ended, eight pseudodifferential or four differential, software selectable

Type of ADC..... Successive approximation

Resolution ..... 12 bits, 1 in 4,096

Max sampling rate..... 100 kS/s

Input signal ranges

Board Gain (Software Selectable)	Board Ranges (Software Selectable)	
	±5 V	0 to 10 V
1	±5 V	0 to 10 V
2	±2.5 V	0 to 5 V
5	±1 V	0 to 2 V
10	±500 mV	0 to 1 V
20	±250 mV	0 to 500 mV
50	±100 mV	0 to 200 mV
100	±50 mV	0 to 100 mV

Input coupling ..... DC

Max working voltage .....In DIFF or NRSE
(signal + common mode).....mode, the negative input/AISENSE should remain within ±5 V of AGND (bipolar), or –5 to 2 V (unipolar). The positive input should remain within the –5 to +10 V range.

Overvoltage protection .....±35 V powered on, ±25 V powered off

Inputs protected .....ACH<0..7>

FIFO buffer size.....4,096 samples

Data transfers .....DMA, interrupts, programmed I/O

DMA mode .....Scatter gather

Dither .....Available

Transfer Characteristics

Relative accuracy.....±0.5 LSB typ dithered, ±1.5 LSB max undithered

DNL .....±1 LSB max

No missing codes.....12 bits, guaranteed

Offset error

Pregain error after calibration.....10 µV max

Pregain error before calibration.....±20 mV max

Postgain error after calibration .....1 mV max

Postgain error before calibration .....±200 mV max

Gain error (relative to calibration reference)

After calibration.....0.02% of reading max

Before calibration .....±2% of reading max

Amplifier Characteristics

Input impedance

Normal powered on .....100 GΩ in parallel with 50 pF

Powered off..... 4.7 kΩ min

Overload..... 4.7 kΩ min

Input bias current ..... ±100 pA

Input offset current..... ±100 pA

CMRR ..... 70 dB, DC to 60 Hz

Dynamic Characteristics

Bandwidth

Small signal (–3 dB)

Gain	Bandwidth
1-10	250 kHz
20	150 kHz
50	60 kHz
100	30 kHz

Settling time for full-scale step

Gain	Settling Time (Accuracy ±0.024% (±1 LSB))
1	10 µs typ, 14 µs max
2–10	13 µs typ, 16 µs max
20	15 µs typ, 19 µs max
50	27 µs typ, 34 µs max
100	60 µs typ, 80 µs max

System noise (*including quantization error*)

Gain	Dither off	Dither on
1–50	0.3 LSB rms	0.5 LSB rms
100	0.5 LSB rms	0.7 LSB rms

Stability

Recommended warm-up time ..... 15 min.

Offset temperature coefficient

Pregain .....±15 µV/° C

Postgain .....±100 µV/° C

Gain temperature coefficient .....±40 ppm/° C

Explanation of Analog Input Specifications

Relative accuracy is a measure of the linearity of an ADC. However, relative accuracy is a tighter specification than a *nonlinearity* specification. Relative accuracy indicates the maximum deviation from a straight line for the analog-input-to-digital-output transfer curve. If an ADC has been calibrated perfectly, this straight line is the ideal transfer function, and the relative accuracy specification indicates the worst deviation from the ideal that the ADC permits.

A relative accuracy specification of ±1 LSB is roughly equivalent to, but not the same as, a ±0.5 LSB nonlinearity or integral nonlinearity specification because relative accuracy encompasses both nonlinearity and variable quantization uncertainty, a quantity often mistakenly assumed to be exactly ±0.5 LSB. Although quantization uncertainty is ideally ±0.5 LSB, it can be different for each possible digital code and is actually the analog width of each code. Thus, it is more specific to use relative accuracy as a measure of linearity than it is to use what is normally called nonlinearity, because relative accuracy ensures that the *sum* of quantization uncertainty and A/D conversion error does not exceed a given amount.

Integral nonlinearity (INL) in an ADC is an often ill-defined specification that is supposed to indicate a converter's overall A/D transfer linearity. The manufacturer of the ADC chip National Instruments uses on the PCI-1200 specifies its integral nonlinearity by stating that the analog center of any code will not deviate from a straight line by more than ±1 LSB. This specification is misleading because, although a particularly wide code's center may be found within ±1 LSB of the ideal, one of its edges may be well beyond ±1.5 LSB; thus, the ADC would have a relative accuracy of that amount. National Instruments tests its boards to ensure that they meet all three linearity specifications defined in this appendix.

Differential nonlinearity (DNL) is a measure of deviation of code widths from their theoretical value of 1 LSB. The width of a given code is the size of the range of analog values that can be input to produce that code, ideally 1 LSB. A specification of ±1 LSB differential nonlinearity ensures that no code has a width of 0 LSBs (that is, no missing codes) and that no code width exceeds 2 LSBs.

System noise is the amount of noise seen by the ADC when there is no signal present at the input of the board. The amount of noise that is reported directly (without any analysis) by the ADC is not necessarily the amount of real noise present in the system, unless the noise is considerably greater than 0.5 LSB rms. Noise that is less than this magnitude produces varying amounts of flicker, and the amount of flicker seen is a function of how near the real mean of the noise is to a code transition. If the mean is near or at a transition between codes, the ADC flickers evenly between the two codes, and the noise is very near 0.5 LSB. If the mean is near the center of a code and the noise is relatively small, very little or no flicker is seen, and the noise is reported by the ADC as nearly 0 LSB. From the relationship between the mean of the noise and the measured rms magnitude of the noise, the character of the noise can be determined. National Instruments has determined that the character of the noise in the PCI-1200 is fairly Gaussian, so the noise specifications given are the amounts of pure Gaussian noise required to produce our readings.

Explanation of DAQ Rates

Maximum DAQ rates (number of S/s) are determined by the conversion period of the ADC plus the sample-and-hold acquisition time, which is specified at 10 µs. During multichannel scanning, the DAQ rates are further limited by the settling time of the input multiplexers and programmable gain amplifier. After the input multiplexers are switched, the amplifier must be allowed to settle to the new input signal value to within 12-bit accuracy. The settling time is a function of the gain selected.

Analog Output

Output Characteristics

Number of channels.....Two voltage

Resolution.....12 bits, 1 in 4,096

Typical update rate .....20 S/s–1 kS/s, system dependent

Type of DAC .....Double buffered

Data transfers .....Interrupts, programmed I/O

Transfer Characteristics

Relative accuracy (INL) ..... ±0.25 LSB typ, ±0.50 LSB max

DNL ..... ±0.25 LSB typ, ±0.75 LSB max

Monotonicity ..... 12 bits, guaranteed

Offset error

- After calibration..... ±0.2 mV max
- Before calibration ..... ±50 mV max

Gain error (relative to internal reference)

- After calibration..... ±0.01% of reading max
- Before calibration ..... ±1% of reading max

Voltage Output

Ranges ..... 0 to 10 V, ±5 V, software selectable

Output coupling ..... DC

Output impedance..... 0.2 Ω typ

Current drive ..... ±2 mA

Protection..... Short circuit to ground

Power-on state..... 0 V

Dynamic Characteristics

Settling time to full-scale range (FSR) .. 5 µs

Stability

Offset temperature coefficient ..... ±50 µV/° C

Gain temperature coefficient..... ±30 ppm/° C

Explanation of Analog Output Specifications

Relative accuracy in a D/A system is the same as nonlinearity because no uncertainty is added due to code width. Unlike an ADC, every digital code in a D/A system represents a specific analog value rather than a range of values. The relative accuracy of the system is therefore limited to the worst-case deviation from the ideal correspondence (a straight line), except noise. If a D/A system has been calibrated perfectly, the relative accuracy specification reflects its worst-case absolute error.

DNL in a D/A system is a measure of deviation of code width from 1 LSB. In this case, code width is the difference between the analog values produced by consecutive digital codes. A specification of ±1 LSB differential nonlinearity ensures that the code width is always greater than 0 LSBs (guaranteeing monotonicity) and is always less than 2 LSBs.