Universidad de Oriente
Facultad de Ingeniería Eléctrica
Departamento de Ingeniería Biomédica

DISEÑO E IMPLEMETACIÓN DE UNA HERRAMIENTA DE SOFTWARE PARA LA SÍNTESIS DE BIOSEÑALES

TESIS PRESENTADA EN OPCIÓN AL TÍTULO DE INGENIERO BIOMÉDICO

Autor: Asiel Aldana Ortíz.
Tutores: Ing. Joan Lambert Cause.
MSc. Alexander A. Suárez León.
Lic. Anmarli Olivia Rodríguez Ferreiro.

Santiago de Cuba
2013

Universidad de Oriente
Facultad de Ingeniería Eléctrica
Departamento de Ingeniería Biomédica

DISEÑO E IMPLEMETACIÓN DE UNA HERRAMIENTA DE SOFTWARE PARA LA SÍNTESIS DE BIOSEÑALES

TESIS PRESENTADA EN OPCIÓN AL TÍTULO DE INGENIERO BIOMÉDICO

Autor: Asiel Aldana Ortíz.
Tutores: Ing. Joan Lambert Cause.
MSc. Alexander A. Suárez León.
Lic. Anmarli Olivia Rodríguez Ferreiro.

Santiago de Cuba
2013

DEDICATORIA

Dedico esta tesis en primer lugar a mis padres, pues es el resultado también de su trabajo por más de 23 años, gracias por todo, nunca podré pagarles tanto, pero tal vez este sea un buen comienzo, los amo.

A mis hermanos, en especial a mi hermana Arelis, por quererme como lo hace.

A mis sobrinitos.

.

AGRADECIMIENTOS

Agradezco a Dios por regalarme estos 5 magníficos años, y estar al lado mío en cada uno de ellos. Agradezco a mis padres, por ser el ejemplo más vivo que tengo de sacrificio, amor y entrega.

A mi hermana Arelis y a mis sobrinitos Daykelito y Ashani.

A mis amigos de todas partes, a mis compañeros de aula y profesores, gracias por hacer de mi quien soy. Un agradecimiento especial a mis tutores por tantas horas de sacrificio. A mi oponente Ledea, por su dedicación y consejos.

A los compañeros de grupo que iniciaron conmigo hace 5 años, a esos que quedaron en el camino y a los que se sumaron, por los buenos y malos momentos que hemos compartido, por todas las risas cuando algo salía bien y las lamentaciones cuando todo salía mal, especialmente a mis compañeros de cuarto: David, Adrian, Renato, Geralexis, Favier,Lázaro, Doboy, Joaquín, Poti; con los cuales he compartido gratos momentos y cuya amistad difícilmente olvide.

A todos Muchas Gracias.

"Dios es el que justifica"
Romanos 8:33

RESUMEN

Los sintetizadores de señales por software han devenido en una alternativa viable a los costosos sistemas de generación de señales por hardware. Este trabajo ha tenido como objetivo, diseñar e implementar una herramienta de software para la síntesis de bioseñales. Para la realización del mismo se utilizó el IDE propuesto por Embarcadero C++Builder2010®. La herramienta implementada permite la generación de patrones de formas de ondas electrocardiográficas, fotopletismográficas, incluyendo otras de referencia como cuadrada, triangular, diente de sierra, sinusoidal y trapezoidal. También permite la manipulación manual de patrones de acuerdo a las necesidades de diseño, posibilita el filtrado digital de señales a través de técnicas basadas en el diseño de filtros IIR y FIR, facilita el análisis espectral a partir del cómputo de la FFT, además de permitir la compatibilidad con diferentes formatos de archivo; entre otras funcionalidades incluidas en el ambiente de trabajo diseñado para la aplicación. Con el objetivo de validar cada uno de los algoritmos implementados, se realizaron pruebas de carácter cualitativo y cuantitativo a la herramienta, arrojando cada una de estas, resultados satisfactorios.

Palabras claves: Sintetizadores de señales, software, FIR, IIR, FFT, generador.

ABSTRACT

Synthesizers software signals have become a viable alternative to the expensive systems for signal generation hardware. This work has aimed to design and implement a software tool for the synthesis of bio-signals. To achieve the same was used IDE proposed Embarcadero C++Builder2010®. Implemented tool allows the generation of patterns electrocardiographic waveforms, photoplethysmographic, including other reference as square, triangular, sawtooth, trapezoidal and sinusoidal. Also allows manual manipulation of patterns according to design requirements, enables digital signal filtering through techniques based on the design of IIR and FIR filters, facilitates spectral analysis from the FFT computation, and allows compatibility with different file formats, among other features included in the work environment designed for the application. In order to validate each of the algorithms implemented, were tested for qualitative and quantitative tool, throwing each of these, results.

Keywords: Signals synthesizers, software, FIR, IIR, FFT, generator.

CONTENIDO PAG

01. Acrónimos y Terminología.

PC	Computadora Personal.
ECG	Electrocardiograma.
PPG	Fotopletismografía.
EEG	Electroencefalograma.
EOG	Electrooculograma.
FFT	Fast Fourier Transform (Transformada Rápida de Fourier).
DDS	Direct Digital Synthesis (Síntesis Digital Directa).
AWG	Arbitrary Waveform Generator (Generador arbitrario de forma de ondas).
DFT	Discrete Fourier Transform (Transformada Discreta de Fourier).
TF	Transformada de Fourier.
DIT	Decimation In Time (Diezmado en el tiempo).
DIF	Decimation In Frequency (Diezmado en frecuencia).
IDFT	Inversa de la Transformada Discreta de Fourier.
DSP	Digital Signal Processing (Procesamiento digital de señales).
RF	Radio frecuencia.
SNC	Sistema Nervioso Central.
CPU	Central Processing Unit (Unidad Central de Procesamiento)

02. Símbolos.

f	Frecuencia.
f_s	Frecuencia de muestreo.
s	Frecuencia compleja.
t	Tiempo.
T_s	Período de muestreo.
ω	Frecuencia angular.
V	Volt.
mV	mVolt.

INTRODUCCIÓN

En sus inicios los sintetizadores de señales eran dispositivos estáticos, apenas configurables, en la actualidad permiten la conexión y control desde una computadora personal (PC), a través de la cual pueden ser manipulados mediante software hechos a la medida según la aplicación, aumentando las prestaciones y la flexibilidad del mismo. Estas aplicaciones están dotadas de algoritmos de procesamiento y generación altamente eficientes con el objetivo de ofrecer un tratamiento lo más detallado posible de la información que se está manipulando. El desarrollo de formas de onda por software ha llegado en los últimos años a través de herramientas profesionales de modelación como por ejemplo *MATLAB*® de la compañía *The Mathworks*®, *LabView*® y *LabWindow*® de la *National Instruments*™. Si bien estas herramientas continúan siendo esenciales para la mayoría de los proyectos de desarrollo, en ocasiones resulta de vital importancia el uso de aplicaciones específicas que ofrezcan características de trabajo que faciliten el mismo.

Hoy en día existe en el mercado internacional una gran cantidad de dispositivos diseñados para la generación de señales por hardware, que van desde los tradicionales generadores de funciones hasta modernos generadores arbitrarios asistidos directamente por computadoras. Cada uno de ellos procedente de fabricantes específicos con características de trabajo particulares para cada tipo, según su complejidad [1].

Compañías especializadas en el diseño de dispositivos para la generación de formas de onda, se empeñan en la actualidad en dotar a sus aplicaciones de modernas herramientas de simulación de señales, con el fin de brindar mayor versatilidad a sus productos. Entre estas se puede citar el caso de la *National Instruments*™ mencionada anteriormente, la cual dispone de un conjunto de herramientas que incluyen desde sofisticados dispositivos de hardware, hasta modernas herramientas de software con un gran número de prestaciones. *BK PRECISION*® es otras de las compañías que ofrece internacionalmente una gran diversidad de dispositivos generadores de señales, capaces de cubrir diversas necesidades, desde los más sofisticados para trabajos de laboratorio hasta módulos para usos educativos. Entre las aplicaciones de software que este distribuye se encuentra el software *Wave Xpress*™ el cual permite a los usuarios crear y editar casi cualquier forma de onda imaginable. Actualmente las aplicaciones de software comerciales permiten además de

importar señales desde osciloscopios, cargarlas desde archivos de texto o formatos específicos, así como crear formas de onda a partir de bosquejos utilizando herramientas de dibujo y edición.

Con el desarrollo de la electrónica y de los sistemas de cómputo, los generadores de señales, han incrementado su presencia en el mercado internacional, con una rápida evolución y un creciente aumento de sus prestaciones técnicas, logrando mayor flexibilidad y compatibilidad con otros dispositivos y sistemas.

Mientras que los generadores de funciones por hardware tradicionales producen ondas sinusoidales, cuadradas y triangulares de frecuencias y amplitudes precisas, los generadores modernos además de las anteriores incluyen capacidades de generar ondas de pulso, trapezoidales y diente de sierra, con características y parámetros variables. Además existen algunos capaces de generar ondas especiales, como simulaciones de terremoto o de señales de ECG (electrocardiograma), facilitando su manipulación a través de herramientas de software diseñadas con tal fin [1]. Es en este último grupo donde entran a jugar un papel fundamental los generadores arbitrarios de formas de ondas, del inglés, *Arbitrary Waveform Generator* (AWG), los cuales permiten crear señales de estímulos sofisticadas que reflejan con alta precisión aplicaciones del mundo real. Los AWG han revolucionado la generación de señales de estímulos al reducir en gran medida la necesidad de construir circuitos especiales para generar señales específicas. Esto puede reducir los costos al eliminar múltiples pasos en los procesos de diseño, construcción y depuración de los mismos.

El desarrollo de generadores de bioseñales por software implicaba que para cada tarea, se requiriera de un dispositivo en particular. Se entiende así que un generador ideal es un generador arbitrario, que pueda manipular un amplio rango de formas de onda, así como también brindar un amplio rango de frecuencias, con excelentes resoluciones en amplitud y fase.

Por razones de carácter económico se dificulta el uso extendidos en centros de estudio de nivel superior en Cuba de estas tecnologías. Lo que da lugar a que el país y por ende sus

universidades carezcan del libre acceso a dichas herramientas, tan necesarias para el estudios en asignaturas vinculadas a la instrumentación y el procesamiento de señales.

Por los motivos antes expuestos, se evidencia la siguiente situación problémica:

- Baja disponibilidad de aplicaciones capaces de generar bioseñales y manipularlas de forma eficiente, pudiendo servir como fuentes de referencia y material de estudio en asignaturas vinculadas al tratamiento y procesamiento de estas.

Atendiendo a lo expuesto anteriormente se plantea el siguiente:

Problema de investigación:

Dificultades en la obtención de formas de ondas digitales ajustadas a las necesidades de trabajo, en las asignaturas vinculadas al tratamiento y procesamiento de bioseñales.

Objetivo general de la investigación:

Diseñar e implementar una herramienta de software para la síntesis y manipulación de señales de uso frecuente como: ECG, PPG entre otras, para las asignaturas vinculadas al tratamiento y procesamiento de bioseñales.

Objeto de investigación:

Sintetizadores de señales por software, específicamente en la generación y procesamiento de señales de origen fisiológico.

Campo de acción:

Sistemas de cómputo para la síntesis de bioseñales.

Hipótesis:

Si se diseña e implementa una herramienta de software para la síntesis de bioseñales, se dispondrá de una aplicación capaz de garantizar la obtención y manipulación de formas de ondas digitales ajustadas a las necesidades de trabajo, en asignaturas vinculadas al tratamiento y procesamiento de señales biomédicas, dando solución al problema de investigación propuesto.

Métodos del nivel teórico:

- **Histórico Lógico:** Para hacer un análisis de los antecedentes, causas, condiciones y evolución de los sintetizadores de señales y su papel dentro de las ciencias biomédicas.
- **Análisis y Síntesis:** Interpretación y valoración de los referentes teóricos de la síntesis de señales por software y de datos obtenidos en el proceso de investigación.

Métodos del nivel Empírico:

Utilizando métodos **empíricos** se ha llegado a la concepción de algoritmos de interés para el proceso investigativo.

Para el desarrollo de la presente investigación se han planteado las siguientes tareas:

- Estudio de las características generales de los sintetizadores de señales por software.
- Análisis de las prestaciones de la herramienta de software C++Builder2010®.
- Recopilación de información sobre los principales métodos y herramientas de generación y procesamiento de señales.
- Diseño de los algoritmos de generación, edición, análisis y procesamiento de señales, propuestos en la investigación.
- Implementación de la herramienta de software.

El presente proyecto propone una herramienta de software para la síntesis de bioseñales implementada bajo el IDE de Embarcadero C++Builder2010®. Empleando además como herramientas de apoyo otras aplicaciones profesionales como MATLAB R2010a®, Proteus® 7, Microsoft Visual C++2010®, Microsoft Office Visio 2007®, Microsoft Office Word 2007®, Microsoft Office PowerPoint 2007®, entre otras, las cuales garantizaron el cumplimiento en tiempo de los objetivos propuestos.

1. CAPÍTULO 1. CONCEPTOS BÁSICOS

En el presente capítulo se describen las características básicas de las principales bioseñales, así como de algunas fuentes de perturbaciones asociadas a las mismas. En este se plantean también los métodos más usados actualmente, para la generación de señales sintéticas por software. Además de hacer referencia a las principales herramientas para el procesamiento y edición de formas de onda.

1.1 El cuerpo como sistema y las señales biomédicas

En el cuerpo humano se desarrollan un conjunto de procesos químicos, electroquímicos, biológicos y fisiológicos encaminados a controlar y mantener la homeostasis. Todos estos procesos son fenómenos complejos, que se acompañan o manifiestan mediante señales que reflejan su naturaleza y actividad. Tales señales pueden ser de diversos tipos, desde señales bioquímicas procedentes de hormonas y neurotransmisores, hasta señales eléctricas, como potenciales o corrientes. Las enfermedades y trastornos en sistemas biológicos causan alteraciones en los procesos fisiológicos normales, trastornando estos procesos y provocando el mal funcionamiento del organismo.

1.1.1 Clasificación de las señales biomédicas

Una señal se define como una función que varía con el tiempo, el espacio o cualquier otra variable o variables independientes. Esta función contiene información, generalmente sobre el estado de un sistema físico. Las señales se pueden representar de diferentes formas, en todos los casos la información estará contenida en la forma de variación de alguno de sus parámetros.

Partiendo del concepto anterior se puede plantear que las señales biomédicas son registros espaciales, temporales o espacio-temporales de eventos tales como el latido del corazón o la contracción de un músculo. La actividad eléctrica, química o mecánica que ocurre durante estos eventos biológicos frecuentemente produce señales que pueden ser medidas y analizadas. En consecuencia las señales biomédicas o bioseñales contienen información que puede ser utilizada para explicar los mecanismos fisiológicos subyacentes en un evento o un sistema biológico específico.

Las bioseñales pueden ser clasificadas de muchas maneras. Algunas de las formas más importantes de clasificarlas son según [2]:

- La fuente o naturaleza física.
- La aplicación biomédica.
- Las características de la señal.

1.1.1.1 Clasificación según la fuente o naturaleza física

En función de su fuente o naturaleza física, es posible distinguir entre bioseñales continuas (están definidas a lo largo de un intervalo continuo de tiempo y son descritas por funciones de variables continuas) o discretas (definidas solo en puntos discretos del tiempo o del espacio y representadas como secuencias de números). Las señales producidas por sistemas biológicos son casi siempre señales continuas. De acuerdo a su fuente de origen las bioseñales se pueden agrupar de la siguiente forma [2]:

- Bioimpedancia
- Bioacústica
- Biomagnética
- Biomecánica
- Bioquímica
- Biooóptica
- Bioeléctrica

- **Señales de Bioimpedancia:**

La impedancia eléctrica de los tejidos contiene información importante sobre su composición, volumen, actividad endocrina, actividad del sistema nervioso autónomo, y más. La señal de bioimpedancia se genera usualmente inyectando en el tejido bajo prueba corrientes sinusoidales de 20 μA a 20 mA y frecuencias entre 50 KHz y 1 MHz [2].

- **Señales bioacústicas:**

Muchos fenómenos biomédicos generan perturbaciones sonoras. La medición de éste provee información acerca del fenómeno que lo produce, un ejemplo clásico pudiera ser el flujo de sangre en el corazón o a través de las válvulas cardíacas, en las cuales se generan sonidos típicos [2].

- **Señales biomagnéticas:**

Varios órganos, como el cerebro, el corazón y los pulmones, producen campos magnéticos extremadamente débiles. La medición de tales campos provee información no incluida en otras bioseñales. Debido al bajo nivel de los campos magnéticos que se tienen que medir, deben tomarse precauciones extremas en el diseño del sistema de adquisición de estas señales [2].

- **Señales biomecánicas:**

Se originan de alguna función mecánica del sistema biológico. Estas señales incluyen aquellas producidas por la locomoción y el desplazamiento, señales de flujo, presión, y otras. La medición de estas señales biomecánicas requiere una gran variedad de transductores, no siempre sencillos y económicos. El fenómeno mecánico no se propaga, como lo hacen los campos magnéticos y eléctricos y las ondas acústicas. Por lo tanto, la medición se tiene que realizar usualmente en el sitio exacto donde se origina [2].

- **Señales bioquímicas:**

Las señales bioquímicas son el resultado de mediciones químicas de los tejidos vivos o de muestras analizadas en el laboratorio clínico. La medición de la concentración de iones dentro y en las vecindades de una célula, por medio de electrodos específicos para cada Ion, es un ejemplo de este tipo de señal. La presión parcial de oxígeno ($P0_2$) y de dióxido de carbono ($PC0_2$) en la sangre o en el sistema respiratorio son otros ejemplos. Las señales bioquímicas son, a menudo, de muy baja frecuencia [2].

- **Señales bioópticas:**

Las señales bioópticas son el resultado de funciones ópticas en sistemas biológicos que ocurren naturalmente o son inducidas para la medición. La oxigenación sanguínea puede estimarse midiendo la luz transmitida y reflejada por los tejidos a distintas longitudes de onda, mediante este método puede obtenerse información importante acerca del feto

midiendo la fluorescencia del líquido amniótico, entre otras aplicaciones. El desarrollo de la tecnología de fibra óptica ha abierto un amplio espectro de estudios de señales bioópticas [2].

- **Señales bioeléctricas:**

La señal bioeléctrica es propia de los sistemas biológicos. Su fuente es el potencial transmembrana, el cual ante ciertas condiciones puede variar para generar un potencial de acción. En mediciones sobre células aisladas, donde se utilizan micro electrodos como transductores, el potencial de acción es en sí mismo la señal biomédica. [2].

1.1.1.2 Clasificación según la aplicación biomédica

La señal biomédica es adquirida y procesada con propósitos de diagnóstico, monitorización o de otro tipo. La clasificación puede ser concebida de acuerdo con el campo de aplicación, por ejemplo cardiología o neurología. Tal clasificación puede ser de interés cuando el propósito es, por ejemplo, el estudio del sistema fisiológico [2].

1.1.1.3 Clasificación según las características de la señal

Desde el punto de vista del análisis de la señal, éste es el método de clasificación más relevante. Cuando el propósito fundamental es el procesamiento, no es relevante cual es la fuente de la señal o a qué sistema biomédico ésta pertenece; lo que es primordial son las características de la señal. En este sentido, las bioseñales pueden ser clasificadas como determinísticas o como aleatorias. Las señales determinísticas pueden ser descritas mediante funciones matemáticas o reglas. Las funciones matemáticas no pueden ser usadas para describir precisamente las señales aleatorias, que exhiben distribuciones probabilísticas y pueden ser expresadas en términos de propiedades estadísticas [2].

1.1.2 Origen y características de los eventos bioeléctricos

Debido a la importancia de este tipo de bioseñales y la generalización que en los últimos años ha tenido el estudio de las mismas, resulta relevante el conocimiento de las características de estas como vía esencial para entenderlas mejor. Aunque las señales bioeléctricas provenientes de las distintas células varían considerablemente en amplitud y forma, todas tienen un origen común en el potencial transmembrana, que es la diferencia de

potencial eléctrico que existe entre el interior y el exterior de la célula. El límite funcional de las células biológicas es una delgada estructura (aproximadamente 10 nm) formada por lípidos y proteínas, llamada membrana celular. Las fuerzas electroquímicas a través de esta membrana regulan el intercambio químico celular. El medio dentro de la célula (plasma) y fuera de ella (líquido intersticial) está compuesto principalmente por agua conteniendo varios iones. La diferencia de concentración de iones dentro y fuera de la célula produce una fuerza de origen electroquímico sobre la membrana. En estado de reposo, las células musculares y nerviosas mantienen un potencial de membrana de alrededor de -60 a -90 mV, con el interior negativo respecto del exterior. [3]. Aunque existen registros bioeléctricos desde principios del siglo XIX (experimentos de Galvani), es en el siglo XXI en donde se han producido los avances más importantes en esta área. Originalmente, los biólogos con cierta especialización en electrónica, eran capaces de fabricar sus propios instrumentos para captar las señales eléctricas provenientes de los seres vivos.

1.1.2.1 Características generales de las señales bioeléctricas

Los potenciales bioeléctricos del cuerpo humano o de cualquier animal raramente son determinísticos. Sus magnitudes varían con el tiempo, incluso cuando todos los factores que las originan están controlados. Los valores de la misma medida pueden variar enormemente entre diferentes individuos aunque estos estén sanos y las condiciones de medición sean las mismas. Esto quiere decir que los valores pueden ser distintos para diferentes personas, aunque sean valores normales en ellos [5].

A continuación se comentan algunos de los valores típicos para diferentes señales bioeléctricas [5]:

SEÑAL	*MAGNITUD*	*ANCHO DE BANDA (Hz)*
ECG (electrocardiograma)	0,5 - 5 mV	0,01 - 250
EEG (electroencefalograma)	5 - 300 mV	DC - 150
EGG (electrogastrograma)	10 - 1000 mV	DC - 1
EMG (electromiograma)	0.1 - 5 mV	DC - 10.000
EOG (electrooculograma)	50 - 3500 mV	DC - 50
ERG (electroretinograma)	0 - 900 mV	DC - 50

Tabla 1.1 *Señales bioeléctricas.*

En la tabla anterior se fijan los valores típicos de amplitud y el ancho de banda que debe tener el equipo de medida. Tomando como ejemplo la señal del ECG, y para evitar deformaciones mayores del 10%, la *American Heart Association* recomienda que el ancho de banda de los equipos destinados a captar el ECG sea al menos de 0,1 a 100 Hz. No obstante, en función de la patología a detectar puede ser necesario trabajar con mayores anchos de banda [5].

1.2 Características particulares de algunas bioseñales

1.3 Características básicas de la señal electrocardiográfica (ECG)

El electrocardiograma es la representación gráfica de la actividad eléctrica del corazón, que se obtiene con un electrocardiógrafo en forma de cinta continua. Por mucho es uno de los exámenes más rápidos y confiables con los que se cuenta a la hora de un diagnostico, un seguimiento de terapia o un pronóstico de enfermedades cardiovasculares. Éste registra una variación de tensión cardiaca con respecto al tiempo. Dicho análisis consiste en la identificación de diferencias en las formas de onda electrocardiográfica entre los registros de pacientes sanos con las de pacientes enfermos [6]. La señal de ECG tiene componentes relevantes solo entre 0.05 Hz y 150 Hz. Los valores de la señal en la piel oscilan en pocos mV (entre unos 0.5 y 10mV como máximo) [6].

1.3.1.1 Puntos característicos, ondas y complejos que componen el ECG

En su estructura cada ciclo de la señal de ECG está compuesto por una serie de ondas: la onda P, el complejo QRS que a su vez es una agrupación de tres ondas (la onda Q, la onda R y la onda S), la onda T y, en muy raras ocasiones, una onda U. Estas ondas son el resultado de los procesos de despolarización que se inician en el nodo sinoauricular (SA) y se propagan por las vías de conducción auriculares y ventriculares. Teniendo en cuenta a su vez que cada despolarización va acompañada de su correspondiente repolarización. En la figura 1.1 se podrá apreciar con claridad el lugar que ocupa cada onda en el ECG.

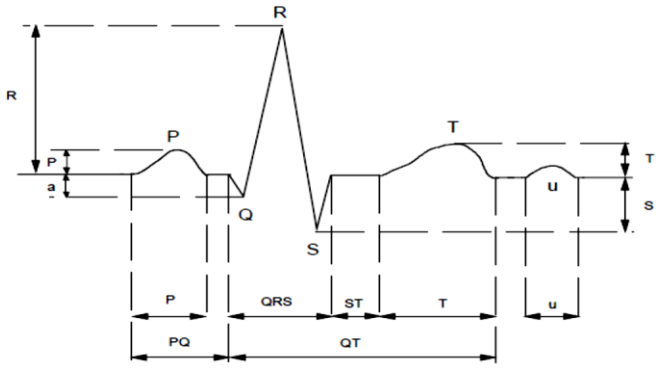


Figura 1.1. *Representación de los tramos característicos de la señal de ECG.*

Onda P:

Esta representa el proceso de excitación auricular.

Onda P-R ó P-Q:

El intervalo P-Q o P-R se mide desde el comienzo de la onda P hasta el final de la onda Q, o el comienzo de la onda R si no existe la onda Q. El segmento P-Q desde el final de P hasta el comienzo de Q representa el retraso en la transmisión del impulso desde las aurículas a los ventrículos por el *Haz de His* [7].

Complejo ventricular QRS:

Está formado por la sucesión de 3 ondas de curso rápido, que representan la despolarización del miocardio ventricular. La duración del complejo QRS se mide desde el comienzo de la onda Q hasta el final de la onda S, dicha duración depende de la edad y la frecuencia cardíaca siendo aproximadamente 0,1 segundos como máximo. En cuanto a la amplitud, se acepta que los límites de la normalidad oscilan entre 0.6 y 2mV [8].

Segmento S-T:

Este segmento, junto con la onda T, representa a las fuerzas originadas en el proceso de recuperación o repolarización ventricular, específicamente este corresponde al lapso comprendido entre la despolarización y la repolarización. Por dicho motivo, debe ser teóricamente isoeléctrico, ya que en ese instante no debe fluir corriente de acción alguna al encontrarse totalmente despolarizada la fibra muscular. En ese estado, todas las cargas son negativas y, lógicamente, no existen diferencias de potencial que genere una corriente [8].

Onda T:

La onda T, junto al segmento S-T, integra los grafo-elementos del proceso de recuperación o repolarización ventricular. El segmento S-T tiene una anchura de 0,10 s a 0,25 s [7].

Onda U:

Es la 6ta onda del electrocardiograma, aunque su presencia no es muy frecuente. Su duración es de aproximadamente 0,16 s a 0,24 s. Su origen no está bien establecido, aunque se supone que corresponde a la activación tardía de algunos sectores del miocardio ventricular [7].

1.3.1.2 Ruido e interferencia asociado a la señal de ECG

El proceso de adquisición de la señal, del electrocardiograma, y en general de las señales bioeléctricas, está afectado por la acción de interferencias de diversas índoles. Como resultado de esta interferencia, la señal que se adquiere puede ser modificada de manera parcial o total [9]. Existen en el medio que nos rodea disímiles fuentes de perturbaciones asociadas a estas bioseñales, entre las más comunes se encuentran:

1) **Derivas de línea base:** Son los desplazamientos suaves de la línea isoeléctrica producto de la respiración que se encuentra en el rango de frecuencias de los 0,15 a los 0,35 Hz. En situaciones de elevado esfuerzo físico durante monitorización ambulatoria o en pruebas de esfuerzo, el ritmo respiratorio aumenta y por consiguiente el movimiento del tórax del paciente y los electrodos colocados sobre este, provocando que las derivas de línea base posean componentes frecuenciales superiores a 0,35 Hz. Al ser de muy baja frecuencia, puede ser minimizada mediante un filtro pasa alto [9].
2) **Interferencia de 60 Hz:** La interferencia de la red eléctrica se manifiesta mediante acoplamiento capacitivo con el sujeto, cables o equipo de monitorización. Su efecto puede ser considerado adicionando a la señal de ECG combinaciones de sinusoides de 60 Hz y sus armónicos fundamentales para diferentes niveles de amplitudes. Para minimizar su efecto, la solución clásica es el filtrado electrónico o por software [9].
3) **Ruido muscular o electromiográfico:** Producto de las contracciones musculares se generan pequeños potenciales con contenido espectral entre la componente de corriente

directa y los 10 kHz. Aunque generalmente son de pequeña amplitud, durante el ejercicio pueden corromper la señal de ECG y provocar un mal funcionamiento de los algoritmos de detección [9].

4) **Artefactos de movimiento:** Su principal causa se debe a la interferencia transitoria producto de la pérdida de contacto, permanente o intermitente, entre el electrodo y la piel del sujeto a causa del movimiento y las vibraciones del mismo. Esta conmutación puede generar grandes artefactos debido a los acoplamientos capacitivos presentes [9].

5) **Ruido electroquirúrgico:** El equipamiento de electrocirugía genera interferencia de banda ancha (0.5 a 3MHz) que puede ser modulada por descargas transitorias que se generan en la propia unidad [9].

1.3.2 Características básicas de la señal de fotopletismografía (PPG)

El principio físico de la fotopletismografía está basado en determinar las propiedades ópticas de un área determinada de la piel [10]. Para esto, se emite luz infrarroja sobre la piel, la cual es absorbida en mayor o menor medida dependiendo de la cantidad de flujo sanguíneo. La luz emitida es reflejada y ésta corresponde con la variación del volumen de sangre. Lo anterior lo podemos visualizar en la figura 1.2.

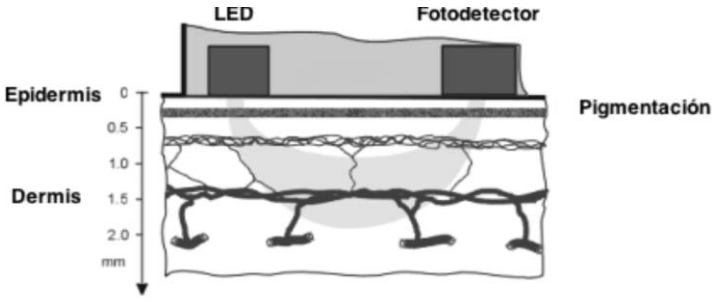


Figura 1.2. *Principio físico de la Fotopletismografía.*

En la figura 1.2 se visualiza el uso de sensores para la captura de las variaciones en el volumen del flujo sanguíneo. Existen técnicas que se utilizan con el mismo propósito, variando la posición o materiales de los sensores. En la figura 1.3 se muestra la forma de onda característica de una señal de fotopletimografía [11].

Figura 1.3. *Forma de onda característica de la señal de Fotopletismografía.*

La figura 1.3 muestra la morfología característica de la señal de fotopletismografía, la cual está estrechamente relacionada con la frecuencia cardíaca. Donde cada elevación en la señal corresponde a una pulsación del corazón. Por ello, la fotopletismografía es una técnica ampliamente usada para realizar diversos estudios encaminados a supervisar la actividad eléctrica del corazón. Entre estos estudios, está el análisis de las variaciones entre las frecuencias cardíacas para medir el nivel de estrés en una persona, entre otras aplicaciones [11].

1.4 Métodos empleados para la síntesis de señales

Un sintetizador de señales, de funciones o de formas de onda por software es una herramienta que genera patrones de señales tanto periódicas como no periódicas. Y se emplea normalmente en el diseño, prueba y reparación de dispositivos electrónicos y de otras aplicaciones de software. Como su nombre sugiere, estas herramientas sintetizan las señales, es decir, las generan a partir de la combinación de elementos simples, normalmente señales periódicas o funciones matemáticas implementadas directamente desde una herramienta de software hecha a la medida [13].

En este apartado se expondrán dos de los métodos que en la actualidad son muy empleados para la síntesis de señales asistida por computadora, con funciones de forma de ondas características tanto periódicas como no periódicas, estos son:

- **Síntesis Digital Directa. (*Direct Digital Synthesis* (DDS)).**
- **Síntesis Aditiva.**

En primer lugar se analizará el método de síntesis digital directa, que no es más que la aplicación directa del teorema de muestreo y es precisamente el método empleado por su efectividad y facilidad de implementación en el presente trabajo. En segundo lugar, se

abordará el método aditivo de síntesis o síntesis aditiva, cuyo principio se basa en la suma de formas de ondas elementales para crear una forma de onda más compleja.

1.4.1 Síntesis Digital Directa

La técnica de síntesis directa está basada en la acumulación de cambios de fase y su reproducción o generación en una forma de onda digitalizada. Esta técnica, también conocida con el nombre de "síntesis de lectura de tabla" es una aplicación directa de la teoría de muestreo. Donde una forma de onda se almacena en la memoria de un ordenador, y haciendo incrementos en la lectura de la misma, se puede direccionar sucesivamente cada punto de la forma de onda y dirigirlo hacia el área de generación o salida, como muestra la figura 1.4 [12]. Este método permite la reconstrucción de una señal a partir de una secuencia uniforme de datos. Conformando una estructura de datos caracterizada generalmente por los siguientes aspectos [13]:

- **Tabla de forma de onda:** Contiene la amplitud en función de la fase para una determinada forma de onda.
- **Acumulador de fase:** Permite configurar el incremento del dato fase de entrada a la tabla de forma de onda.
- **Constante de frecuencia:** Número de pasos de fase entre una entrada a la tabla y la siguiente [13].

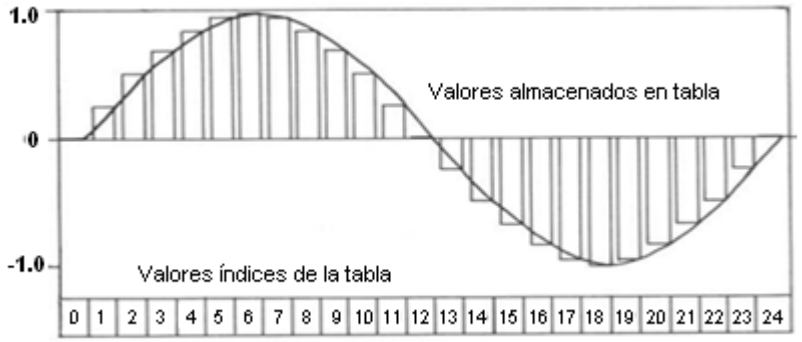


***Figura 1.4**. Síntesis por lectura de tabla.*

1.4.1.1 Generadores de síntesis muestreada

El método empleado para la síntesis desarrollado en el presente proyecto se basa como se planteaba anteriormente en la síntesis digital directa cuyo principio está sustentado sobre la estructura de los generadores de síntesis muestreada, a continuación se enuncian algunos aspectos necesarios a tener en cuenta para el uso e implementación de esta técnica.

- **Cambio de la base de tiempos.**

A partir de cambios en la base tiempo de la señal almacenada en memoria, se podrá generar una secuencia de señal basada en los períodos de tiempo que sean definidos. Dando la posibilidad de crear bases de frecuencia en un rango limitado únicamente por la frecuencia de muestreo de la señal almacenada. A continuación se muestra un ejemplo que ilustra con más claridad el planteamiento anterior [13].

Si se tiene un acumulador de fase igual a 1 y una constante de frecuencia igual a la longitud de la tabla: Se recuperan todos los datos del periodo de la forma de onda, obteniéndose de esta manera una resolución máxima, con un incremento en la lectura de la tabla de una muestra por cada iteración, como se muestra en la figura 1.5 para una tabla de 4000 puntos.

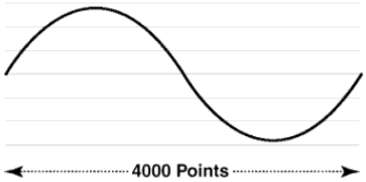


***Figura 1.5**. Síntesis por lectura de tabla 1Hz.*

Del mismo modo para un acumulador de fase igual a 4: Se extrae de la tabla un valor de cada cuatro posibles. De esta forma aumenta la frecuencia y disminuye la resolución de la señal de salida, manteniéndose el número total de puntos para una constante de frecuencia fija igual al tamaño de la tabla, como se muestra en la figura 1.6 [13].

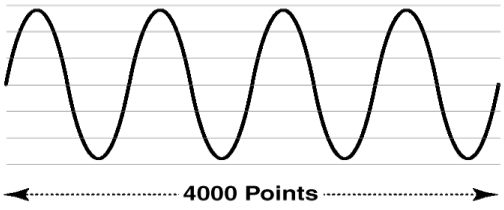


***Figura 1.6**. Síntesis por lectura de tabla 4Hz.*

- **Ventajas:**
  - Gran velocidad de cambio de frecuencia.
  - Elección arbitraria de la referencia de fase.
  - Posible mejora de la resolución y la exactitud.

- **Inconvenientes:**
  - Rango limitado de frecuencias (0... 10MHz).
  - Desventajas propias de un sistema de muestreo.

- **Especificaciones:**
  - Resolución vertical. Número de bits empleados para el almacenamiento de la señal en memoria.
  - Resolución horizontal. Número de bits del acumulador de fase.
  - Profundidad de memoria. Al aumentar el tamaño de la memoria se mejora de la calidad de la señal y se reduce la distorsión.
  - Rango de frecuencia. Limitado por la frecuencia de muestreo y la profundidad de la memoria.

- **Modos de adquisición de la forma de onda:**
  - Biblioteca de formas de onda.
  - Arreglos digitales.
  - Ecuación.
  - Generación con herramientas gráficas.
- **Edición de formas de onda:**
  - Edición gráfica.

- Operadores algebraicos.

1.4.1.2 Generadores Arbitrarios de Forma de Ondas por software (AWG)

Estos tipos de generadores son un caso particular de los sistemas de DDS enfocados generalmente para usos en aplicaciones de instrumentación. Los AWG modernos disponen de herramientas internas para la creación de formas de onda basadas en [13]:

- Definición de la señal punto a punto, por tramos, mediante interpolación o a mano alzada (mediante el mouse).
- Definición mediante fórmulas matemáticas.
- Importación de señales del mundo real adquiridas mediante osciloscopios digitales o sistemas de adquisición de datos afines con la aplicación.
- Definición de las señales en el dominio de la frecuencia.

Aplicaciones de los AWG.

Los AWG proporcionan la capacidad de realizar pruebas de anomalías en una gran variedad de aplicaciones como pudieran ser.

- Filtrado de señales.
- Adición de Ruido.
- Análisis frecuencial de señales.
- Simulación de latidos en señales ECG, PPG, EEG entre otras.

Los AWG en la Educación.

Constituyen una alternativa de bajo coste a toda una variedad de generadores complejos y de precio elevado como pudieran ser:

- Generadores de funciones.
- Generadores de audio.
- Generadores de tonos.
- Generadores de pulsos.
- Generadores de patrones de señales.

Alternativa al uso en simulaciones por ordenador:

- Se obtiene una señal similar a la del mundo real que puede ser visualizada, analizada y manipulada de forma totalmente real e intuitiva.

En resumen esta técnica da una amplia capacidad de creación de señales al usuario, mediante herramientas aportadas por aplicaciones especializadas en la generación arbitraria de formas de ondas.

1.4.2 Síntesis aditiva

La síntesis aditiva es uno de los primeros métodos utilizados para la obtención de espectros sonoros con características similares al comportamiento natural de los sonidos. El análisis acústico de las señales de audio define que los sonidos naturales están, de hecho, compuestos de una multitud de componentes simples, las cuales, en el caso de los sonidos periódicos con una altura determinada, tienen una frecuencia múltiplo de la frecuencia fundamental. El principio esencial de este método se basa en realizar operaciones elementales (sumas, multiplicaciones) en formas de ondas simples (senos, cosenos), con el objetivo de obtener una señal con las características morfológicas esperadas. Haciendo uso eficiente de métodos de modelación matemática como las series de Fourier [12].

1.5 Análisis y procesamiento de bioseñales

En la etapa de procesamiento se intenta destacar la información deseada del resto de la señal, que usualmente tiene ruido asociado, y de la cual puede interesar tan sólo una parte (eliminación de señal de electromiografía (EMG) superpuesto al ECG, por ejemplo). En este caso se aplican técnicas de atenuación y cancelación de ruido (filtrado digital, adaptativo, promediado), para lo cual se necesita un conocimiento previo de las características de la señal y del ruido asociado a las mismas para cada caso [14].

En algunas aplicaciones el ruido tiene características frecuenciales (espectro) que no interfieren, o al menos no significativamente, con el espectro de la señal. Además, estas características no varían con el tiempo. En este caso, podemos utilizar un filtro digital, que elimine las componentes frecuenciales del ruido preservando las de la señal. La figura 1.7

muestra el caso de una señal de ECG con interferencia procedente de la red (50 Hz) y la misma señal después de ser filtrada.

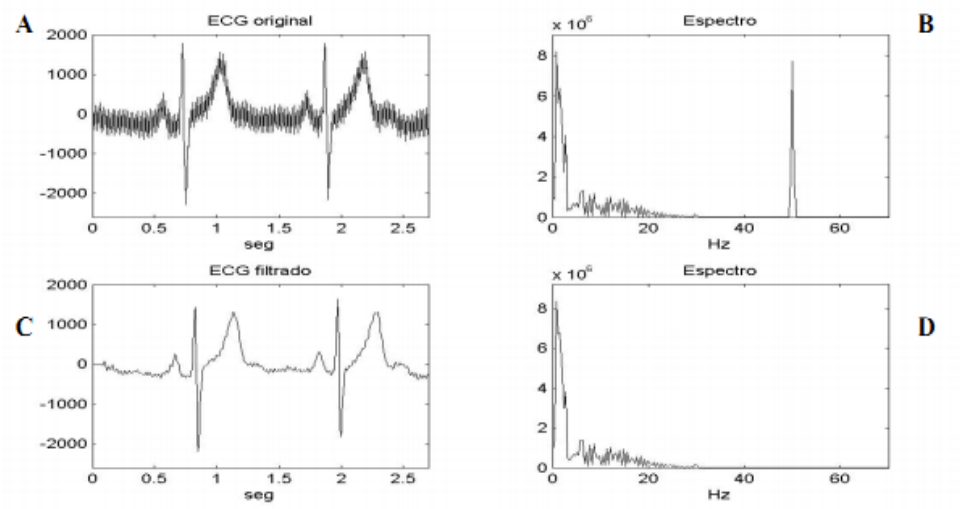


Figura 1.7. *ECG con interferencia de red (A) y su espectro (B), ECG filtrado elimina-banda (C) y el espectro correspondiente (D: puede observarse la eliminación de la frecuencia de red), tomado de [14].*

1.5.1 Filtros digitales, conceptos básicos

Los filtros digitales presentan algunas ventajas sobre los correspondientes analógicos. En primer lugar, un filtro digital es más inmune al ruido debido a la forma en que se implementa (software o circuitos digitales). La precisión depende sólo del error de redondeo, que está determinado por el número de bits utilizado en el diseño para representar las variables del filtro. Además, las características de este (frecuencia de corte, orden, entre otras.) pueden cambiarse fácilmente, y su funcionamiento es independiente del entorno (variaciones de temperatura, tensión de alimentación, entre otras.). Esta característica es importante en aplicaciones médicas donde las señales tienen frecuencias bajas que pueden ser distorsionadas por las derivas del circuito analógico [14].

Un filtro digital se puede definir como un sistema discreto que transforma una secuencia de entrada produciendo la correspondiente secuencia de salida. Si la transformación no es función del tiempo, será un sistema invariante temporal, y si cumple el principio de superposición lineal, se denominará lineal. El presente trabajo centra su interés sobre alguno de los filtros más comunes, como es el caso de los invariantes temporales y lineales [14].

Dada una secuencia de muestras, {x(n)}, la respuesta del filtro a dicha entrada se puede expresar como:

$$y(n) = \sum_{k=0}^{N-1} h(k) * x(n-k) \quad [1.1]$$

donde $h(k)$es la respuesta del filtro al impulso, y el producto entre $h(k)$ y las muestras de la señal retardada, representan la convolución entre dicha respuesta al impulso y la señal de entrada. Se dice que el filtro es estable si la respuesta del mismo a una entrada finita es también finita, y es causal si no produce salida hasta que se aplica señal a la entrada (es decir, y(n)=0 para n<0).

La relación entre entrada y salida del filtro suele expresarse de forma más eficiente mediante la transformada Z. Su definición para una secuencia de entrada $x(k)$ viene dada por:

$$X(z) = \sum x(k) * z^{-k} \quad [1.2]$$

siendo z una variable compleja. Una propiedad importante de la transformada Z es que la transformada de la convolución de dos secuencias es equivalente al producto de las respectivas transformadas. Si se aplica a la expresión [1.1], se obtiene:

$$Y(z) = H(z) * X(z) \quad [1.3]$$

donde H(z), es la función de transferencia del filtro. Si evaluamos H(z) en el plano z para $z=e^{j\omega nT}$ en los puntos de la circunferencia de radio unidad, obtenemos la función compleja H(ω), que representa la respuesta en frecuencia del filtro. En función de dicha respuesta, los filtros pueden clasificarse como pasa-bajos, pasa-altos, pasa-bandas o supresor-bandas. La figura 1.8 muestra la respuesta ideal para cada tipo. La expresión general de la función de transferencia está dada por [1.4]:

$$H(z) = \frac{\sum_{m=0}^{M-1} b_m z^{-m}}{1+\sum_{k=1}^{N-1} a_k z^{-k}} \quad [1.4]$$

donde los ceros y polos de la función serán las raíces del numerador y denominador respectivamente. A partir de esta expresión puede obtenerse la ecuación en diferencias en el dominio temporal:

$$y(n) = -\sum_{k=1}^{N} a_k * y(n-k) + \sum_{m=1}^{M} b_m * x(n-m) \qquad [1.5]$$

que corresponde a una forma recursiva puesto que la salida depende tanto de muestras de entrada como de salidas anteriores. Cuando todos los coeficientes a_k son cero, la salida del filtro depende exclusivamente de las muestras de entrada, y corresponde a una forma no recursiva.

Una clasificación usual de los filtros digitales se basa en su respuesta al impulso. Cuando $h(k)$ presenta un número finito de valores no nulos, se obtiene una respuesta finita al impulso (filtros FIR: Finite Impulse Response), mientras que si $h(k)$ presenta siempre valores no nulos, se obtiene una respuesta infinita (filtros IIR: Infinite Impulse Response). Este último caso requiere algún tipo de realimentación (forma recursiva), que puede influir en la estabilidad del filtro. Los filtros FIR, que suelen implementarse usualmente de forma no recursiva y son estables, pueden diseñarse para que tengan respuesta de fase lineal, lo que evita distorsiones en la señal de salida [14].

El diseño de filtros se realiza en base a ciertos requerimientos, generalmente relacionados con su respuesta en frecuencia. Puesto que las respuestas ideales presentadas en la figura 1.8 no son realizables, se implementan aproximaciones con un error aceptable.

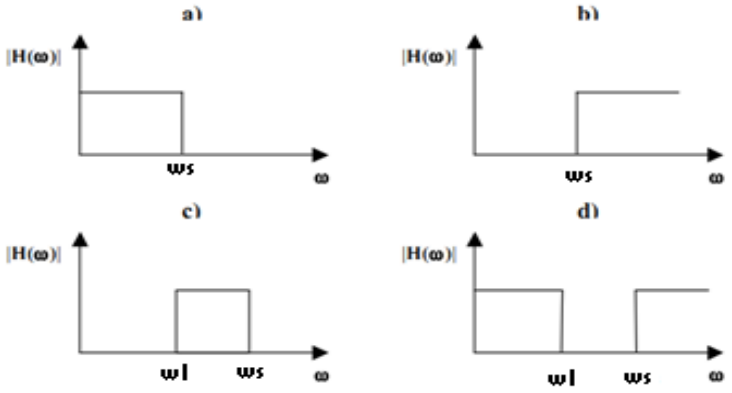


Figura 1.8 *Módulo de la respuesta en frecuencia para filtros ideales: a) pasa-bajo, b) pasa-alto, c) pasa-banda, d) elimina-banda. ωI y ωs: frecuencias de corte inferior y superior, respectivamente.*

Existen diversas técnicas de diseño, entre las que podemos destacar el método de ventanas, el muestreo en frecuencia y el diseño de rizado constante para el caso FIR, y el método del impulso invariante o la transformación bilineal para filtros IIR. Un ejemplo usual de aplicación de filtrado digital para señal ECG es el caso de detección de complejos QRS. El objetivo es destacar las frecuencias asociadas a dichos complejos atenuando los de las demás ondas (P, T). Se ha aplicado un filtro pasa-banda FIR de fase lineal, y su módulo y fase así como la señal de entrada y su correspondiente salida filtrada se muestran en la figura 1.9. Como puede observarse, es posible establecer un umbral de amplitud en la señal filtrada que sólo corte en puntos de los QRS, lo que no era factible en el caso de la señal de entrada ya que las ondas T presentan amplitudes similares a los QRS [14].

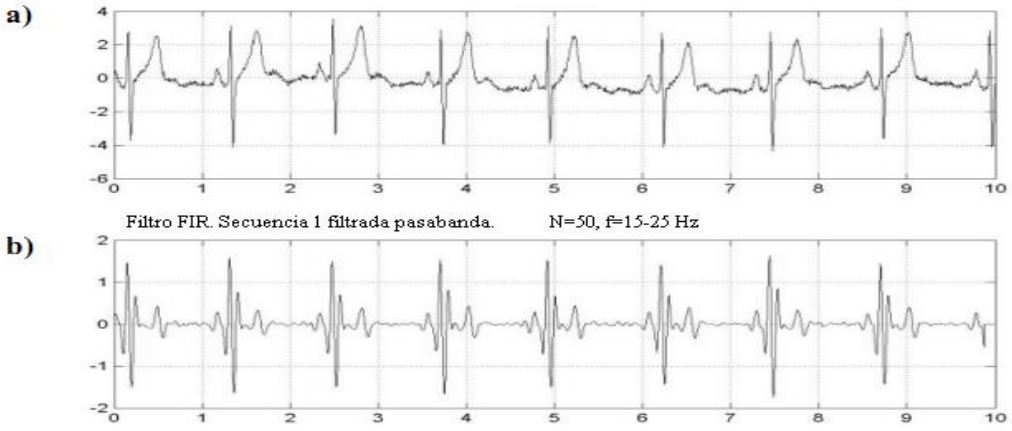


Figura 1.9 *Filtrado pasa-banda FIR aplicado a señal ECG: a) Señal de entrada: b) Salida filtrada (15-25Hz).*

1.5.2 Análisis frecuencial

Las bioseñales pueden ser representadas tanto en el dominio temporal como en el dominio de la frecuencia. Las componentes de la frecuencia de una señal, que pueden resultar difíciles de discernir en una representación temporal, pueden ser separadas y analizadas más fácilmente en el dominio de las frecuencias. Para muchos métodos y aplicaciones de procesamiento de diferentes bioseñales, incluidos el filtrado y el análisis espectral, es útil tener la representación de las bioseñales en el dominio de la frecuencia. A lo largo de los años la transformada de Fourier (TF) ha jugado un papel protagónico en el análisis frecuencial de señales discretas. El análisis de Fourier se dedica al estudio de señales:

periódicas o no periódicas, continuas o discretas, en el dominio del tiempo, o de cualquier otra variable unidimensional, bidimensional o multidimensional [16].

Resumiendo la TF es una representación de una función en el dominio de la frecuencia que contiene exactamente la misma información que la señal original y sólo difiere en la manera en que esta es representada.

1.5.2.1 Dominio discreto de Fourier

La Transformada Discreta de Fourier (DFT del inglés Discrete Fourier Transform) es el equivalente discreto de la Transformada de Fourier donde se ha transformado la variable continua 't' por la variable discreta 'n*Ts' siendo 'Ts' el periodo de muestreo. Planteando que la Transformada de Fourier de una señal analógica x(t) es:

$$X(\omega) = \int_{+\infty}^{+\infty} x(t) * e^{-j\omega t} dt \qquad [1.6]$$

La Transformada Discreta de Fourier es un método muy eficiente para determinar el espectro en frecuencia de una señal. Permite convertir una secuencia de valores en el dominio del tiempo a una secuencia de valores equivalente en el dominio de la frecuencia. La inversa de la Transformada Discreta de Fourier (IDFT) realiza el proceso contrario. A continuación se muestran el par de ecuaciones que caracterizan tanto a la DFT como a su inversa (Ecuación. 1.7 y 1.8 respectivamente) [16]:

$$X(k) = \sum_{n=0}^{N-1} x(n) * W^{nk} \qquad para\ k = 0,1,\dots,(N-1) \qquad [1.7]$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) * W^{-nk} \qquad para\ n = 0,1,\dots,(N-1) \qquad [1.8]$$

Donde las constantes 'W' son conocidas como factores *twiddle* y definidas como:

$$W = e^{-j2\Pi/N} \qquad [1.9]$$

Uno de los objetivos de este proyecto ha sido el de implementar un algoritmo eficiente que realice la DFT. El algoritmo propuesto se basa en la Transformada Rápida de Fourier (FFT del inglés *Fast Fourier Transform*). Un algoritmo para la FFT obtiene los mismos

resultados que la DFT pero con mayor rapidez debido a que reduce el número de cálculos requeridos para realizar la misma.

El término genérico Transformada Rápida de Fourier abarca distintos algoritmos con distintas características, ventajas y desventajas. Entre los más importantes se encuentran:

- Algoritmo de Goertzel.
- Diezmado en el tiempo. (DIT del inglés Decimation In Time).
- Diezmado en fecuencia. (DIF del inglés Decimation In Frequency).
- Algoritmo de Don Cross

1.5.2.2 Transformada Rápida de Fourier (FFT)

En la fórmula de la Transformada Discreta de Fourier obtener X(k) para un 'k' determinado requiere aproximadamente N sumas complejas y N productos complejos, ya que:

$$X(k) = x(0) + x(1) * W^{k} + x(2) * W^{2k} + x(3) * W^{3k} + \cdots x(N-1) * W^{(N-1)k} \qquad [1.10]$$

para k = 0, 1, .., N-1. Si lo que se desea es obtener X(0), X(1), ..., X(N-1) entonces se necesitarán un total de aproximadamente N^2 sumas complejas y N^2 productos complejos. Esto quiere decir que los requerimientos computacionales de la DFT pueden ser excesivos especialmente si el tamaño de N es grande [16].

1.6 Herramienta de desarrollo Embarcadero C++Builder2010®

Embarcadero C++Builder2010® es un entorno de desarrollo rápido de aplicaciones (RAD, Rapid Application Development) para Windows. Mediante esta herramienta de programación se pueden realizar aplicaciones Win32 de consola (tipo DOS), de interfaz gráfica de usuario (GUI, Graphical User Interface) o cliente-servidor con una gran rapidez y eficiencia. El entorno de desarrollo integrado (IDE, Integrated Development Environment) propuesto por Embarcadero C++Builder2010® consta de las herramientas necesarias para diseñar, implementar, ejecutar y depurar una aplicación, mediante una serie de elementos que operan de forma integrada y complementaria: un editor de código, un depurador de errores, una barra de herramientas, herramientas de bases de datos, entre otras opciones.

Actualmente los sistemas informáticos por software deben ser eficaces y efectivos, además de ofrecer una interfaz de fácil uso en entornos de programación que van sufriendo muchos cambios adaptándose a los tiempos de producción e implantación que son cada vez más cortos [17]. El lenguaje de programación C es uno de ellos, convirtiéndose en uno de los más usado por los ingenieros de sistemas para el desarrollo de software en ámbitos como la industria y la investigación principalmente, por lo que presenta una ventaja dada la portabilidad que ofrecen las aplicaciones desarrolladas desde el mismo, en diferentes sistemas operativos y diversas arquitecturas de computadoras además de estar estandarizado en sus versiones bajo una misma base de instrucciones y reglas de programación [18].

La evolución natural del lenguaje C ha sido C++, que llegó a reducir tiempos de desarrollo de software por el hecho de contar con una arquitectura basada en la llamada programación orientada a objetos, en donde los objetos son, en esencia, componentes de software reutilizables que modelan elementos del mundo real, con los que se crean módulos de programa que realizan un proceso específico y por lo cual C++ se ha convertido en el lenguaje dominante dentro de la implementación de sistemas [18].

En las aplicaciones actuales es de especial importancia que la interfaz de usuario facilite el trabajo con el programa, haciéndolo no sólo mas fácil y accesible, sino también más cómodo y agradable. Los sistemas operativos que cuentan con interfaz gráfica se componen de objetos, funciones y mensajes que se pueden utilizar por medio de un lenguaje de programación orientada a objetos como el propuesto por el IDE de C++Builder2010®, consiguiendo un ahorro de trabajo por la incorporación de jerarquía de clases incluidas dentro de la estructura de un código ya escrito y probado para usarse por medio de las herramientas RAD (herramientas de desarrollo rápido) o lenguajes visuales, que se basan en el uso de componentes o controles prefabricados que son puestos en ventanas y personalizados mediante propiedades haciendo más fácil su utilización mediante el mouse, eventos de teclados entre otras formas de manipulación soportada por el sistema operativo empleado, lo que permite al programador centrarse en el núcleo del programa, que es específicamente el encargado de dar el tratamiento adecuado a la información manipulada.

La programación visual propuesta por el IDE de Embarcadero C++Builder2010® está orientada a lo que sucede en la interfaz, al tratarse de un entorno gestionado por eventos, por lo que no es el propio programa el que establece el flujo de ejecución sino que éste viene determinado por las acciones externas del usuario o algún otro elemento externo.

Para la programación del software propuesto en la presente investigación es necesario la utilización de un lenguaje de programación que no solo permita la manipulación y tratamiento de los datos, sino que además pueda gestionar un diseño óptimo de cada una de las estructuras que se interrelacionan en la misma, con el objetivo de crear una interfaz agradable e intuitiva. Por lo dicho anteriormente y dada las facilidades y técnicas de trabajo que nos ofrece el IDE propuesto por Embarcadero RAD Estudio 2010 en su aplicación C++Builder2010®, resulta esta la opción óptima para convertirse en la plataforma de desarrollo principal en el diseño e implementación de la aplicación a diseñar en el presente proyecto.

2. CAPÍTULO 2. DISEÑO E IMPLEMENTACIÓN

La síntesis de señales por software es un proceso cuya complejidad depende de los tipos de señales con las que se pretenda trabajar. Las aplicaciones más complejas, además de generar las señales, realizan operaciones de filtrado, cálculos estadísticos, medición de magnitudes físicas, análisis espectrales en línea, entre otras, por lo que resulta de vital importancia entender que el empleo adecuado de las herramientas y técnicas de cálculo que se utilicen en el diseño de una aplicación de este tipo, juega un papel fundamental en el resultado final de la misma. En el presente capítulo se describirán cada uno de los métodos y algoritmos empleados en el desarrollo de este proyecto con el objetivo de facilitar la comprensión de los mismos.

2.1 Herramientas para la síntesis y edición de señales

En este apartado se describirán los procedimientos y funciones que han sido diseñados para la implementación de cada uno de los algoritmos propuesto, específicamente de aquellos asociados a los módulos de generación y edición de señales.

2.1.1 Algoritmos de síntesis por lectura de tablas.

Según el algoritmo propuesto para la síntesis de señales en la presente investigación, para modificar la frecuencia de la señal obtenida, debemos en principio variar la velocidad de lectura de la tabla donde esta está almacenada. Por ejemplo, si una tabla de 8192 posiciones la leemos a una tasa de muestreo de 8192 Hz, obtendremos una frecuencia de 1 Hz. Si la leemos a 16384, la frecuencia será de 2 Hz. Si la frecuencia de muestreo es fija, tenemos que proceder mediante saltos de muestras o repetición de muestras para variar la frecuencia de la señal de salida. El algoritmo más elemental de lectura de una tabla de longitud "L" puede ser realizado en un proceso en dos tiempos que se expresa como sigue:

$$index_{fase} = modL\left(fase_{precedente} + incremento\right) \quad [2.1]$$

$$salida = amplitud * tabla[index_{fase}] \quad [2.2]$$

La primera etapa contiene una suma y una operación módulo. El módulo recorta a la longitud L de la suma, en el caso en que esta sea mayor que L. De esta manera, el índice de la fase ($index_{fase}$) es siempre inferior o igual a la longitud de la tabla. La segunda etapa lee el valor de la tabla situado en la posición "$index_{fase}$" y multiplica el valor obtenido por un factor de amplitud. La frecuencia obtenida depende entonces directamente del valor del incremento. De esta forma siendo "fs" la frecuencia de muestreo y "fd" la frecuencia deseada. Para obtener diferentes frecuencias, será necesario variar el incremento según la relación siguiente:

$$incremento = \frac{fd * L}{fs} \quad [2.3]$$

Es decir,

$$fd = \frac{incremento * fs}{L} \quad [2.4]$$

Por ejemplo, si deseamos una frecuencia de 20 Hz y disponemos de una tabla de 2048 valores con una frecuencia de muestreo de 8192 Hz, el incremento valdrá 5. Por lo tanto, leeremos de la tabla una muestra de cada 5. (Ejemplo. Posición [0, 5, 10,15,…]). Después de la muestra a la posición 2045, comenzamos por el principio de la tabla por la posición 2. De hecho, [(2045+5) modulo 2048] =2050−2048 = 2 [12].

2.1.1.1 Problemática de la lectura sin interpolación

Si la frecuencia de referencia que se desea obtener corresponde a un incremento no entero, aparecerá un ruido en la lectura. Este sería un caso corriente, y significaría que la posición real de lectura en la tabla estaría entre dos posiciones. Cuando las tablas son muy grandes, esta situación no es catastrófica. Pero si el redondeo es muy pronunciado, obtendremos diversos artefactos en la señal resultante. La solución a este problema es utilizar un método de lectura con interpolación. Este método es más costoso en tiempo de cálculo, pero da resultados muy satisfactorios para casi todas las longitudes de tabla. La interpolación implica utilizar la parte fraccional del incremento para calcular el valor preciso de la muestra misma si es que ésta no está presente en la tabla. Este proceso se lleva a cabo

leyendo dos muestras de la tabla (el correspondiente al valor entero del incremento y el siguiente) e interpolando entre estos dos valores (ver ejemplo en la figura 2.1) [12].

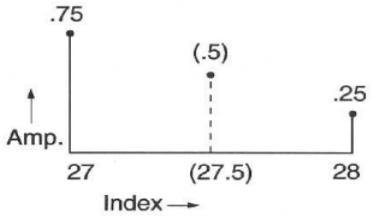


Figura 2.1. *Síntesis por lectura de tabla interpolada .*

2.1.2 Generación de formas de ondas

Antes de continuar con la descripción de los elementos principales que compones la etapa de generación de forma de onda se deben de tener en cuenta las características y funcionalidades que pudiera tener dicha etapa, con el propósito de establecer los límites y vínculos de esta con el resto de la aplicación desarrollada. Debido a que el objetivo central de este proyecto se enmarca en el diseño e implementación de una herramienta que facilite la generación de patrones no solo de bioseñales, sino también de otras morfologías, es preciso tener en cuenta los parámetros y opciones que cada una de estas es capaz de aportar. A continuación se citan las distintas formas de ondas que fueron consideradas para la implementación en el proceso de diseño de la presente investigación, a través de la herramienta principal que se diseña, estas son:

- Forma de onda de señal de ECG.
- Forma de onda de señal de PPG.
- Forma de onda sinusoidal.
- Forma de onda cuadrada.
- Forma de onda triangular.
- Forma de onda diente de sierra.
- Forma de onda trapezoidal.

2.1.2.1 Forma de onda de la señal de ECG y PPG

Debido a que los modelos de diseños propuestos para la generación de las señales tanto de ECG como de PPG (ver figura 2.2) son idénticos, en este apartado se describirá el mismo sin hacer referencia a ninguna de estas dos formas de ondas de manera particular. Para la generación de estos modelos se parte inicialmente de un fichero en cuyo contenido está

definida la estructura morfológica de la señal en cuestión (ECG_01_1000.dat ó FOTO_01_1000.dat) , este archivo almacena solo un período de señal, cuyos valores han sido previamente normalizados y muestreados a 1000 Hz, Basado en el método de síntesis por lectura de tablas visto en el **epígrafe 2.1.1** se podrá modificar morfológicamente las características de esta señal, tanto en frecuencia como en amplitud. Dando la posibilidad al usuario de modificar arbitrariamente parámetros como:

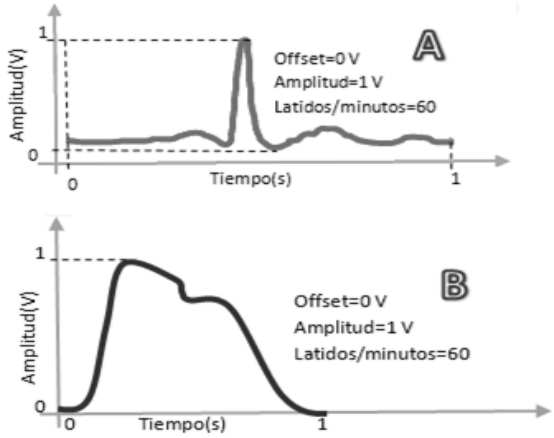


***Figura 2.2**. (A) Señal de ECG, (B) Señal de PPG.*

- Amplitud.
- Número de latidos/minuto.
- Offset.
- Generar variabilidad temporal en cuanto a la ocurrencia de cada latido.

El algoritmo general propuesto se basa inicialmente en generar la forma de onda seleccionada a partir del método de DDS mencionado anteriormente, modificando la frecuencia de cada latido diezmando en mayor o menor medida la señal almacenada en memoria. Si además de esto se conoce que esta señal se encuentra normalizada, se puede proceder a la multiplicación directa de esta por un escalar de amplitud definido por el usuario. A partir de este punto ya se cuenta con un período de señal listo para ser generado, solo queda por parte del usuario precisar el resto de los parámetros que definan la cantidad de latidos que tendrá la misma y el desplazamiento (*offset*). El algoritmo propuesto da además la posibilidad de variar la ocurrencia temporal entre cada latido, de igual forma

diezmando en mayor o menor medida cada período individual de la señal obtenida, a continuación (figura 2.3) se presenta el diagrama de flujos del presente algoritmo.

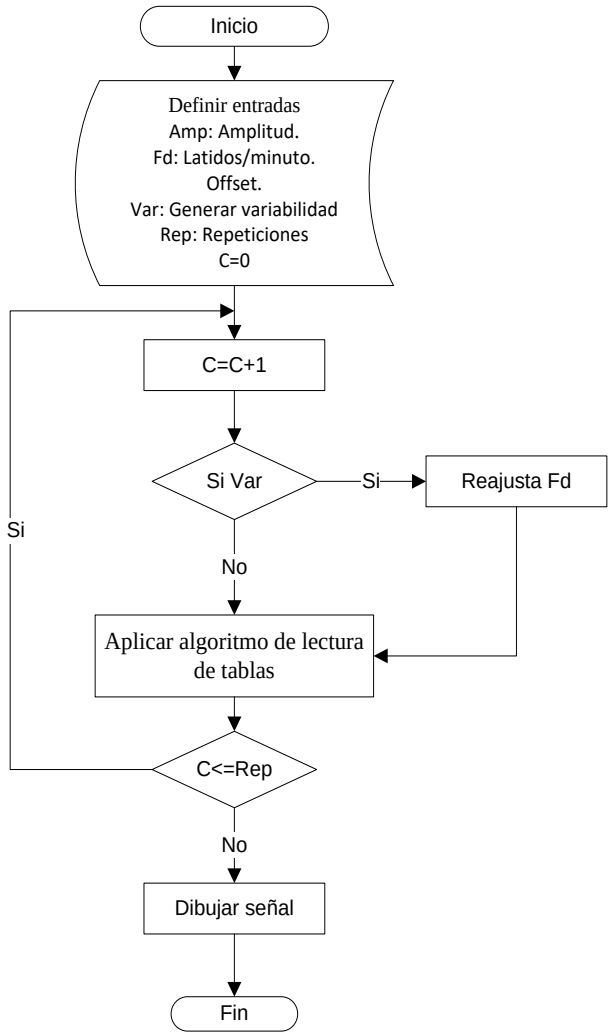


Figura 2.3. *Diagrama de flujos de generación de forma de onda de ECG y PPG.*

2.1.2.2 Forma de onda sinusoidal

Para el dibujo de la onda sinusoidal se hizo uso de la función **sin(*ω*)** propuesta por el IDE de Embarcadero C++Builder2010®, disponible esta desde la librería **math.h**. La misma en su forma simple es capaz de retornar valores reales de doble precisión en un rango entre -1 y 1 (ver figura 2.4).

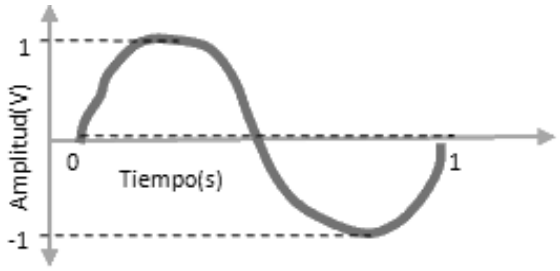


Figura 2.4. *Morfología de onda sinusoidal.*

El parámetro de estrada de esta función queda definido por la siguiente expresión:

$$\omega = 2\pi f_d T_s + fase(Grados) \quad [2.5]$$

De esta forma la función general estará condicionada por la expresión:

$$Y = Amp * sin(2\pi f_d T_s + fase) + offset \quad [2.6]$$

Donde:

- **f_d:** Define la frecuencia deseada.
- **T_s:** Define el período de muestreo de la señal.
- **fase:** Define la fase de la señal resultante representada en grados.
- **Amp:** Define la amplitud de la señal.
- **offset:** Define el desplazamiento en el eje vertical.

De esta forma cada uno de los parámetros citados anteriormente podrán ser modificados directamente por el usuario. El diagrama de flujo para el algoritmo de generación de ondas sinusoidales es el que se presenta en la figura 2.5.

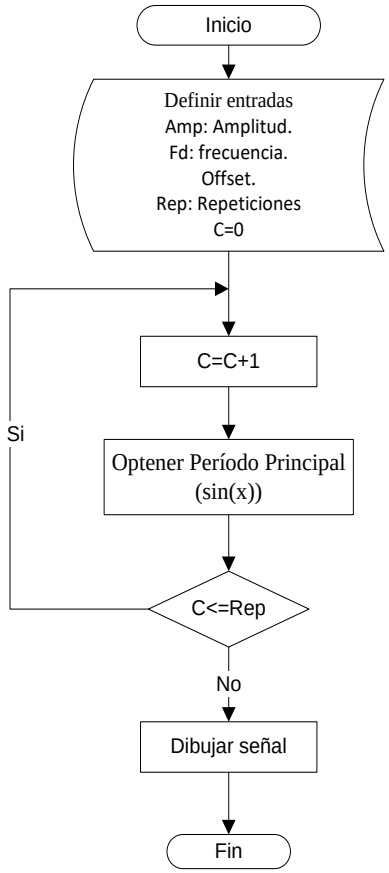


Figura 2.5 *Diagrama de flujo de generación de onda sinusoidal.*

2.1.2.3 Forma de onda cuadrada

La función cuadrada (ver figura 2.6) en la aplicación se genera a partir de un método que dada su morfología pasará del valor de "offset" al valor "*amplitud +offset*" cada cierto tiempo, determinado este por el ciclo de trabajo (*frecuencia de muestreo*) y por la frecuencia deseada en la señal.

Figura 2.6*. Morfología de onda cuadrada.*

El diseño del presente algoritmo centra su funcionamiento inicialmente en la obtención de cada uno de los parámetros de entrada, a partir de los cuales se definirán las características morfológicas de la señal generada, basada en la forma de onda estándar. La mayoría de estos parámetros son definidos en el prototipo de función implementada.

```
void Cuadrada (float Amp, float offset, float frec, float dutycicle, int rep);
```

de esta forma el usuario tiene la posibilidad de modificar desde la pantalla correspondiente de la aplicación, cada uno de los siguientes parámetros:

- **Amp**: Amplitud de la señal, definida por un valor real de simple precisión.
- **Offset**: Define el desplazamiento de la señal en el eje vertical
- **Frec**: Define la frecuencia deseada para la señal obtenida.
- **Dutycicle**: Define en porciento el ciclo útil de la señal de salida.
- **Rep**: Define el número de pulsos a generar (período principal).

Así como también tiene la posibilidad de modificar el nivel de partida en la señal generada. El cuerpo del algoritmo propuesto se resume esencialmente en tres bucles, dos para crear un período y uno para si es necesario repetir la secuencia obtenida. Con el ciclo de trabajo se sacará a partir de qué valor del tamaño del buffer se ha de conmutar de estado alto a estado bajo o viceversa. De los dos bucles que se tienen para rellenar el periodo de valores, el primero irá desde el inicio hasta el valor de conmutación y se incluirán dentro de esas posiciones del buffer el valor *amplitud+offset*. El segundo bucle irá desde el valor de conmutación que ha dado el ciclo de trabajo hasta el final del periodo, y se rellenará con valores iguales al *offset*. El tercer bucle definirá la cantidad de repeticiones que tendrá el período principal, anteriormente generado, en la señal de salida, la figura 2.7 muestra el diagrama de flujo del algoritmo propuesto anteriormente.

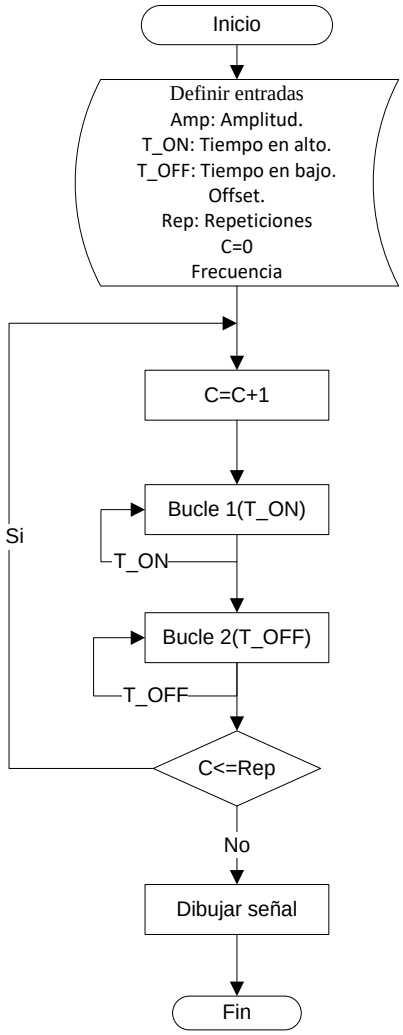


***Figura 2.7**. Diagrama de flujo de generación de onda cuadrada.*

2.1.2.4 Forma de onda triangular

El modelo de diseño propuesto para la implementación del presente algoritmo es similar al de la función cuadrada mostrado anteriormente, pero ahora cambiando la manera de rellenar el buffer de salida. Antes se dividió el periodo en dos tramos, el alto y el bajo, y cada tramo se rellenaba mediante un bucle con los valores deseados. En esta función también se dividirá el periodo en dos tramos. El de subida y el de bajada. Para establecer los límites entre los mismos, se dividirá el tiempo de subida entre el período de la señal a generar (ver expresión 2.8). Ese será precisamente el parámetro de cambio entre los dos bucles como se hizo en la onda anterior (ver figura 2.8). Esta vez para

rellenar el buffer en el primer bucle, se hará sumando el *offset* con un incremento que se calculó con la expresión 2.7:

$$Incremento = \frac{Amplitud}{Tiempo\ de\ subida} \quad [2.7]$$

Para rellenar la segunda parte del periodo lo que se hará es, desde el valor *offset+amplitud*, ir restando un decremento calculado por la expresión 2.9:

$$Período\ = Tiempo\ de\ subida + Tiempo\ de\ caída \quad [2.8]$$

$$decremento = \frac{Amplitud}{Período\ - Tiempo\ de\ subida} \quad [2.9]$$

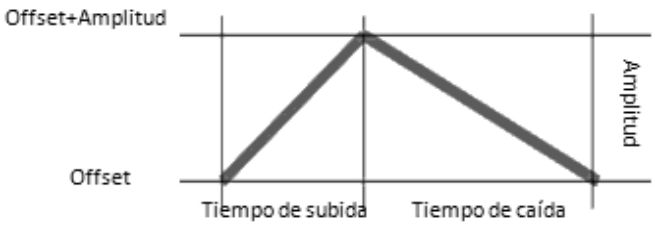


Figura 2.8. *Morfología de onde triangular.*

El prototipo de la función señalada es la que se muestra a continuación:

```
void Triangular(float Amp, float offset, float t_s, float t_c, int rep).
```

El usuario tiene la oportunidad de modificar en este método los siguientes parámetros:

- Amplitud (Amp).
- Tiempo de subida (t_s).
- Tiempo de caída (t_c).
- Offset (offset).
- Número de repeticiones del período principal (rep).

En la figura 2.9 se muestra el diagrama de flujo del algoritmo implementado para la generación del modelo de forma de onda triangular.

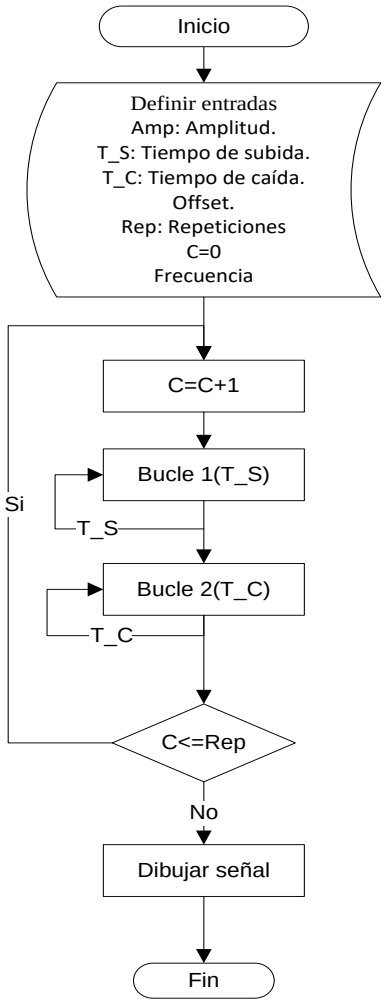


***Figura 2.9**. Diagrama de flujo para la generación de señal triangular.*

2.1.2.5 Forma de onda diente de sierra

Este caso es similar al de la señal triangular, pero en lugar de haber dos bucles para rellenar el buffer, sólo hay uno. Según como se quiera la forma diente de sierra, lo que se hará es sumar el offset más un incremento (amplitud/período) o restar del valor amplitud+offset un decremento (amplitud/período). Así solo se conseguirá la subida o bajada de la onda triangular, que será el diente de sierra deseado. En la figura 2.10 se puede ver dicha función en sus dos formas (ascendente o descendente).

$$Período = Tiempo\ de\ subida\ (Tiempo\ de\ caída) \qquad [2.10]$$

$$Incremento(Decremento) = \frac{Amplitud}{Período} \quad [2.11]$$

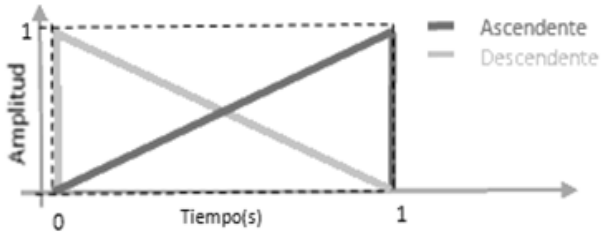


Figura 2.10. *Morfología de onda diente de sierra.*

El prototipo de la función diente sierra señalada es la que se muestra a continuación:

```
void Diente_Sierra(float Amp, float offset, float time, int rep)
```

Para el caso particular de esta señal el usuario tendrá la oportunidad de modificar desde el software los siguientes parámetros:

- Amplitud (`Amp`)
- Tiempo de subida ó caída (`time`)
- Offset (`offset`)
- Número de repeticiones del período principal (`rep`).

El diagrama de flujo del presente algoritmo es el que se presenta en la figura 2.11.

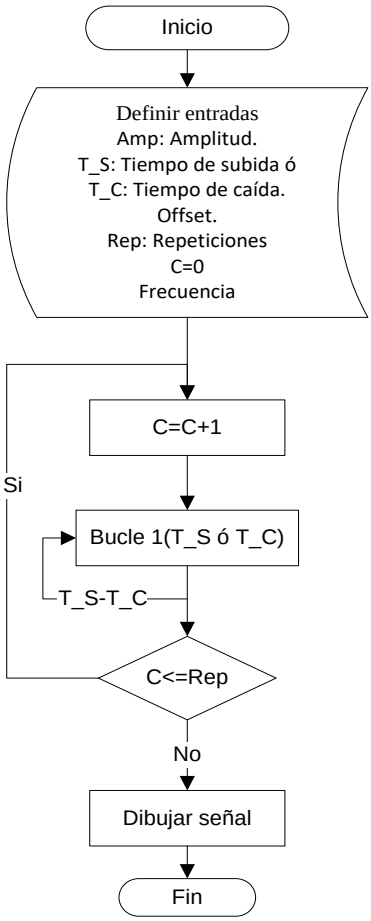


Figura 2.11. *Diagrama de flujo para la generación de señal diente de sierra.*

2.1.2.6 Forma de onda trapezoidal

Este caso es similar a los métodos anteriores solo que el buffer se rellenará mediante 4 bucles. Para dividir el periodo total en 4 partes lo que se hará es, inicialmente, dividir el tiempo de subida entre el periodo total de la señal a generar: la segunda parte quedará definida como el valor de la primera parte, más la división del tiempo en ON entre el periodo de la señal: la tercera parte será el valor de la segunda parte más la división del tiempo de bajada entre el periodo: y la cuarta parte será el periodo restante.

Una vez que se tienen los cuatro bucles definidos se rellenará el buffer como se hizo en la función cuadrada y triangular. La primera parte será el tiempo de subida (T_S) y se rellenará desde *offset* hasta *offset+amplitud* con un incremento calculado por la expresión 2.7, la segunda será el tiempo en alto (T_ON). Es decir, constantemente el valor

offset+amplitud: la tercera parte será el tiempo de caída (T_C). Se rellenará el buffer con valores de *offset+amplitud* menos el decremento que se calcula mediante la expresión 2.9. Y la cuarta parte (T_OFF) será el tiempo en bajo que se rellenará con el valor de *offset (ver figura 2.12)*. En el caso que uno de los tiempos sea nulo implicará que dos valores de partición de bucle coincidan y por lo tanto, esto implica que uno de los bucles no entrará a funcionar y no se formará esa parte. En la figura 2.13 se muestra el diagrama de flujo para la generación de la forma de onda trapezoidal. El prototipo de este método es el que se muestra a continuación:

```
void Trapezoidal(float Amp, float offset, float t_s, float t_c, float
t_ON, float t_OFF, int rep)
```

El usuario tendrá la oportunidad de modificar desde el software los siguientes parámetros:

- Amplitud(`Amp`)
- Tiempo de subida(`t_s`)
- Tiempo en nivel alto(`t_ON`)
- Tiempo de caída (`t_c`)
- Tiempo en nivel bajo(`t_OFF`)
- Offset (`offset`)
- Número de repeticiones del período principal (`rep`).

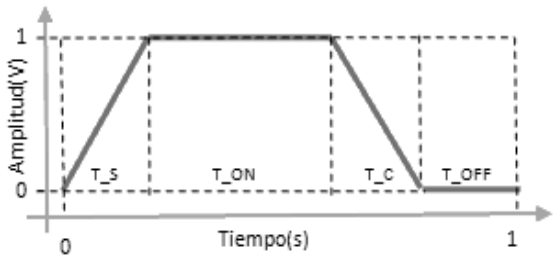


***Figura 2.12**. Morfología de onda trapezoidal.*

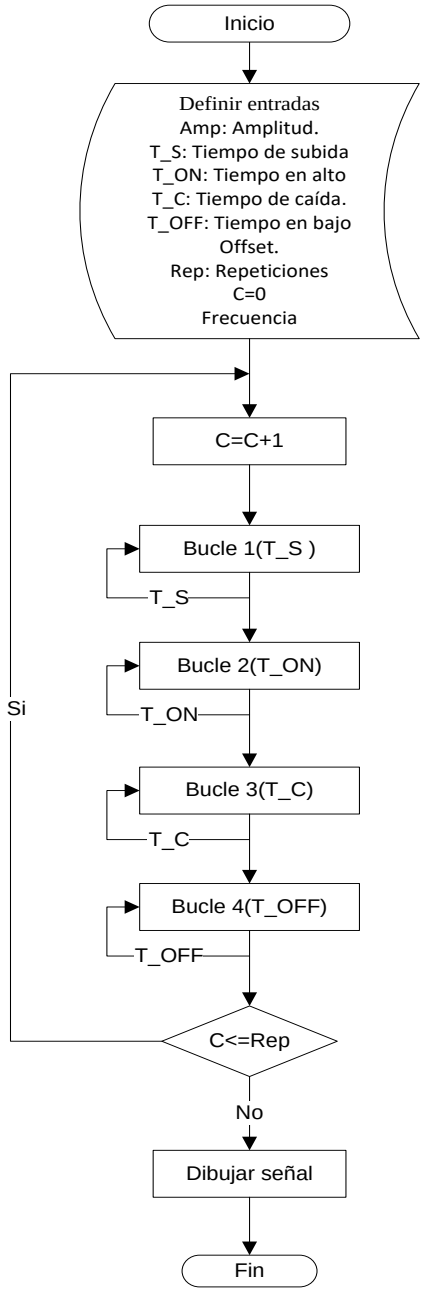


Figura 2.13. *Diagrama de flujo para la generación de señal trapezoidal.*

2.1.3 Herramientas adicionales

Las aplicaciones profesionales que en la actualidad se comercializan en el mercado a nivel internacional, además de brindar una gran diversidad en cuanto a la morfología de las señales que proponen, estas optan por aportarle mayor versatilidad a sus herramientas, incorporándole nuevas prestaciones como:

- **Dibujo manual de formas de onda.**
- **Adición de ruido o interferencia.**
- **Eliminación de tramos a consideración del usuario.**

1. Dibujo manual de formas de onda.

Esta es una de las opciones que en mayor medida facilita el diseño desde la propia aplicación de patrones de forma de ondas con características morfológicas ajustadas a las necesidades del propio usuario. Ya que desde esta se puede modificar o crear de forma manual cualquier señal imaginable. El algoritmo implementado con tal fin en este proyecto sustenta su principio en el empleo de métodos de interpolación, específicamente el método de interpolación lineal, con el objetivo de garantizar a través de una variante de cálculo eficiente, cierta organización en la estructura de la señal manipulada desde el espacio de trabajo que brinda la herramienta de software propuesta.

En general el algoritmo planteado parte de dos puntos, (x_a, y_a) y (x_b, y_b), para obtener un tercer punto interpolado (x, y) a partir de la expresión 2.12 que describe el comportamiento de una función lineal cualquiera, determinada por los puntos anteriormente mencionados:

$$y = y_a + (x - x_a)\frac{(y_b - y_a)}{(x_b - x_a)} \qquad [2.12]$$

Bajo este mismo principio se puede plantear que para un tramo de recta definido por los puntos anteriores, haciendo un incremento en la variable independiente definido por Δx, se podrá interpolar punto a punto sobre cualquier porción de esta, independientemente de su longitud (ver figura 2.14). De esta forma partiendo de un arreglo de solo 2 puntos obtenemos una recta cuyo número máximo de muestras estará determinado por el período de muestreo (Δx) considerado por el usuario, el diagrama de flujo del algoritmo propuesto es el que se muestra en la figura 2.15.

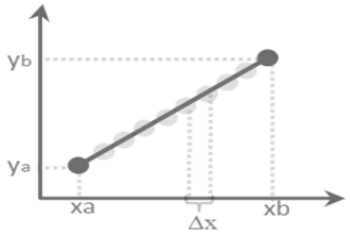


Figura 2.14. *Dibujo de señal por interpolación lineal*

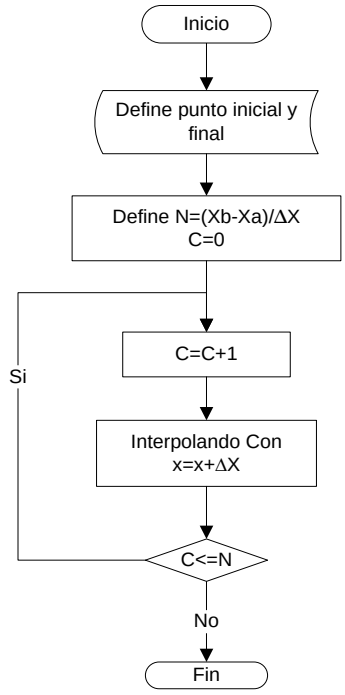


***Figura 2.15**. Diagrama de flujo de la etapa de dibujo manual.*

2. Adición de ruido o interferencia.

En la generación de señales sintéticas para estudios, resulta de vital importancia el hecho de incorporar a las mismas, elementos adicionales con el objetivo de que la morfología obtenida sea capaz de describir con mejor precisión el comportamiento físico de estas, así como la influencia que ejerce el entorno sobre las mismas. Tomando este aspecto especial importancia al referirnos a señales de corte biológico. En las cuales inciden de forma negativa un gran número de factores, favoreciendo a la degradación y pérdida de la información útil contenida en las mismas. De esta forma lo funcional sería que desde nuestra aplicación de software se pudiera modelar el comportamiento de este tipo de fenómenos y tener disponible de esta manera señales con morfologías más cercanas a las reales. Con tal fin se establecen en la implementación del software sintetizador, algunos elementos para la incorporación, tanto de ruido como de interferencias desde la propia aplicación, estos son:

- Ruido blanco gaussiano.
- Interferencia de la red de distribución eléctrica.

- **Ruido blanco gaussiano.**

Para la implementación de este método se hizo uso de la función **rand()** propuesta por el IDE de C++Builder2010® en la librería **stdlib.h.** A partir de esta resulta muy sencillo generar números aleatorios enteros en un rango de hasta 2^{32} elementos. El algoritmo diseñado se basa en generar una cantidad definida de números aleatorios con la función citada anteriormente, a continuación se procede con una etapa de normalización de la trama de datos obtenida multiplicándose finalmente cada elemento del arreglo por un factor de amplitud configurado por el usuario, de este forma se procede a realizar una suma puntal de cada elemento del vector de ruido obtenido con la señal a la cual será incorporado, la figura 2.16 muestra el diagrama de flujo del algoritmo citado anteriormente.

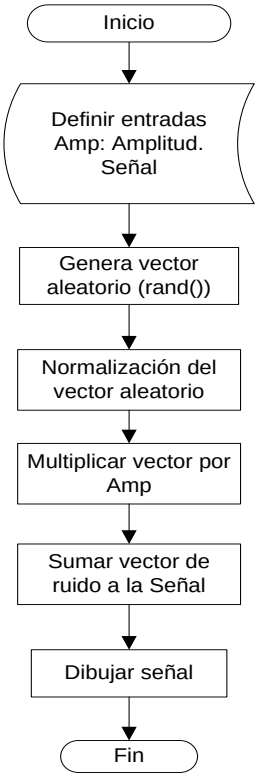


Figura 2.16. *Diagrama de flujo del algoritmo de ruido aleatorio*

- **Interferencia de la red de distribución eléctrica.**

Para la obtención de una morfología de onda que de cierta forma simulara el comportamiento de la señal de interferencia de la red se empleó la función **sin()** incluida

dentro del IDE de C++Builder2010® en la librería **math.h** mencionada anteriormente en el apartado 2.1.2.2**.** Con la simple manipulación de los parámetros de esta función se puede obtener de manera eficiente una forma de onda prácticamente idéntica a la generada por la línea de distribución eléctrica. Dando la posibilidad al usuario de modificar los parámetros de Amplitud, frecuencia y offset de la misma.

3. Eliminación de tramos a consideración del usuario.

Este método da la posibilidad al diseñador de señales de recortar tramos de la misma que no sean de interés práctico a la hora de generar una forma de onda determinada. Así como limitar el tamaño de una señal resaltando solo tramos válidos de esta. El algoritmo básico implementado se desarrolla en dos tiempos, inicialmente el usuario tiene la posibilidad de seleccionar específicamente sobre la señal los rangos límites (mínimo y máximo respectivamente) y posteriormente proceder a aplicar el recorte, con las opciones correspondientes.

2.2 Herramientas para el análisis y procesamiento de señales

El Procesamiento Digital de Señales (DSP), es un área interdisciplinaria de la electrónica, las matemáticas y la informática fundamentalmente, que permite realizar cualquier función de dispositivos electrónicos analógicos y hasta funciones que no pueden ser realizadas con estos, en un sistema digital, normalmente basado en microprocesadores. Con la utilización del DSP se puede realizar cualquier labor de una manera más precisa, ya que los componentes no requieren ajustes y no cambian sus características con el tiempo.

Dentro del análisis y procesamiento digital de señales juegan un papel muy importante empleo de filtros digitales y el aporte que brinda el estudio frecuencial de señales. Por lo que se propone incorporar en la aplicación diseñada un conjunto de herramientas básicas, encaminadas a facilitar estas tareas.

2.2.1 Filtros de coeficientes constantes

Existen dos tipos de filtros de coeficientes constantes, implementables en el dominio del tiempo que deben sin dudas ser citados a la hora de llegar a analizar algún tipo de señal, estos son los Filtros de Respuesta a Impulso Finita (FIR) y los Filtros de Respuesta a Impulso Infinita (IIR).

2.2.1.1 Filtros de Respuesta a Impulso Finita (FIR)

Los filtros FIR, también conocidos como Filtros Transversales (*Transversal Filters*) son los filtros de coeficientes constantes más sencillos de diseñar e implementar. Estos son llamados de respuesta finita, porque el proceso de filtrado se realiza por medio de la convolución de la señal de entrada, con la respuesta al impulso del filtro.

$$H(z) = \sum_{n=0}^{N-1} b_n z^{-n} \qquad [2.13]$$

La función de transferencia de los filtros FIR está dada por (2.13), como se puede observar este filtro solo cuenta con ceros y ningún polo, lo que garantiza la estabilidad del mismo. Es importante destacar que los filtros FIR brindan las siguientes ventajas sobre los filtros IIR:

- Estabilidad garantizada.
- Fase lineal.
- Fácil de diseñar e implementar.

La principal desventaja de los filtros FIR en comparación a los filtros IIR, es el gran número de coeficientes que se necesitan para obtener las mismas características entre uno y otro en la banda de paso y de rechazo.

Propuesta de diseño.

La metodología de diseño de filtros FIR propuesta, se basa en el método de ventanas. Cuyo principio se sustenta en la obtención de los coeficientes de un filtro ideal procediendo posteriormente a la multiplicación o ponderación de dichos valores por los coeficientes de una función denominada función ventana. En la ecuación (2.14), se muestra el cálculo de dichos coeficientes.

$$h(n) = h_d(n) * w(n) \qquad para \ \ -N/2 \leq n \leq N/2 \qquad [2.14]$$

Donde $h(n)$ son los coeficientes del filtro; $h_d(n)$ son los coeficientes del filtro ideal; y $w(n)$ son los coeficientes de las ventanas, en el anexo #4 se muestran las expresiones para

el cálculo de los coeficientes de los filtros ideales y de las ventanas implementadas en el presente trabajo.

Empleo de ventanas.

El truncamiento de las series de Fourier produce filtros FIR con oscilaciones indeseables en la banda de paso y en la banda de rechazo, las cuales resultan de la lenta convergencia de este tipo de series. Para reducir estas oscilaciones, una clase particular de funciones son usadas para modificar los coeficientes de Fourier (respuesta impulso), estas son llamadas ventanas, el truncamiento de las series infinitas de Fourier es equivalente a la multiplicación de los coeficientes con a función ventana.

Las ventanas sirven para suavizar el paso entre los primeros coeficientes del filtro (ambos extremos) y cero, ya que cualquier paso abrupto crea distorsiones armónicas. Una vez obtenido los coeficientes tanto de la función ventana como los de los filtros ideales, se procede a aplicar el filtrado de la señal en cuestión, el conjunto de instrucciones implementados para realizar el filtrado antes mencionado se basa en la función ("*filter(b,a,x)*", implementada en MATLAB®):

```
y=filter(b,a,x)
```

para filtros FIR el parámetro "*a"*=1, en "*b"* se definen los coeficientes anteriormente obtenidos (a partir de expresión 2.14), en x se define el vector de la señal y en "*y*" se obtiene la señal filtrada. Basado en este principio se implementó en lenguaje C++ la función que realiza la convolución lineal de la señal de entrada con los coeficientes del filtro como lo propone la expresión 1.1. A continuación se presentan las instrucciones de códigos implementadas con tal fin.

```
// Aplicando filtro a la señal generada
      for (i=0; i<NBuffer; i++)  // Número de muestras de la señal
      {
            r = 0;
            for (k=0; k<N; k++)
                  r = r + x [i+k] * b [k];   //Convolución lineal
            y[i]= r;                         //Señal filtrada
      }               //donde b[] define coeficientes del filtro.
                      //x[] define muestras de la señal de entrada.
```

2.2.1.2 Filtros de Respuesta a Impulso Infinita (IIR)

Los filtros IIR, también conocidos como Sistemas Auto-regresivos (Auto-Regresive), son llamados de respuesta infinita, porque el proceso de filtrado se realiza por medio de la evaluación de una ecuación en diferencias finitas que regulan un sistema similar al mostrado en la ecuación 1.5. Como la ecuación de diferencias depende de las salidas anteriores del filtro, existe una dependencia de los infinitos estados anteriores de la variable de salida a la variable de salida actual, por tal razón son llamados Filtros de Respuesta a Impulso Infinita.

$$H(z) = \frac{\sum_{m=0}^{M-1} b_m z^{-m}}{1 + \sum_{k=1}^{N-1} a_k z^{-k}} \qquad [2.15]$$

La función de transferencia de los filtros IIR está dada por (2.15), como se puede observar este filtro cuenta con ceros y polos, por lo que la estabilidad del mismo, no está garantizada. Es importante destacar que los filtros IIR brindan una gran ventaja sobre los filtros FIR, esta es la pequeña cantidad de coeficientes que se requieren para obtener una gran atenuación y estrecha banda de transición, en comparación con un FIR de características de salida similares. Por lo que el costo computacional asociado a la ejecución de los mismos tiende a resultar más bajo.

Las desventajas de los filtros IIR en comparación a los filtros FIR son: Problemas de estabilidad, complejidad en el diseño, y fase no lineal.

Propuesta de diseño.

El procedimiento de diseño de los filtros IIR utilizado, está basado en el método de transformada bilineal empleado en la obtención de las expresiones matemáticas para el cálculo de las aproximaciones de Butterworth. En este método a partir de las expresiones de un modelo equivalente analógico, se aplica una transformación de variables para obtener los coeficientes del filtro digital. Esta transformación de variables es conocida como Transformación Bilineal, ecuación (2.16).

$$s = \frac{1 - z^{-1}}{1 + z^{-1}} \qquad [2.16]$$

Además de la transformación de la variable (s->z), es importante también ajustar las frecuencias del plano de Laplace al plano Z, este ajuste (no lineal), es conocido en Inglés como *Frequency Prewarping*, ecuación (2.17).

$$\Omega = \tan(\frac{\omega}{2}) \qquad [2.17]$$

Con esta información, se pueden modificar las formulas de diseño de Filtros Butterworth para diseñar directamente filtros digitales. Para mejorar la sensibilidad de los filtros IIR a la hora de realizar el cálculo para un número finito de muestras, la implementación de los mismos es mejor realizarla como la cascada de varias secciones de bicuadraticas, (ver ecuación 2.18).

$$H(z) = \frac{B_0 + B_1 z^{-1} + B_2 z^{-2}}{1 + A_1 z^{-1} + A_2 z^{-2}} \qquad [2.18]$$

Donde el ángulo de los polos de la función de transferencia de los filtros Butterworth viene dado por la ecuación (2.19), donde N es el orden del filtro.

$$\Phi_{\mathrm{i}} = \frac{\Pi}{2N}(N - 1 + 2i) \quad para\ i = 1, 2, \dots, N \qquad [2.19]$$

Las expresiones empleadas para el cálculo de los coeficientes de este tipo de filtro se relacionan en el anexo #3.

2.2.2 Algoritmo para la FFT basada en DIF

En este proyecto se implementará para el cálculo de la FFT un algoritmo basado en el método de diezmado en frecuencia (DIF) escogido este, debido a la eficiente optimización de código que permite su implementación, hecho que contribuye a la rapidez computacional a la hora de hacer uso del mismo.

El algoritmo para la FFT basada en DIF, descompone la DFT de N puntos en transformadas más pequeñas. Una DFT de N puntos es descompuesta en dos DFT de (N/2) puntos. Cada DFT de (N/2) puntos se descompone a su vez en dos DFT de (N/4) puntos y así sucesivamente. Al final de la descomposición se obtienen (N/2) DFT de 2 puntos cada una. La transformada más pequeña viene determinada por la base de la FFT. Para una FFT de

base 2, N debe ser una potencia de 2 y la transformada más pequeña es una DFT de 2 puntos.

Una vez que se ha llegado a transformadas de dos puntos, ya no se puede seguir descomponiendo y lo que queda es aplicar directamente a cada una de ellas la fórmula de la transformada para N=2 (ver expresión 2.20).

$$X(k) = \sum_{n=0}^{1} x(n) * W^{nk} \qquad k = 0,1 \qquad [2.20]$$

Mediante este algoritmo la secuencia de salida se obtiene de forma desordenada por lo que habrá que reordenarla (ver figura 2.17, para una muestra de salida de 8 elementos), para ello se aplica un algoritmo conocido como '*bit-reversal*'.

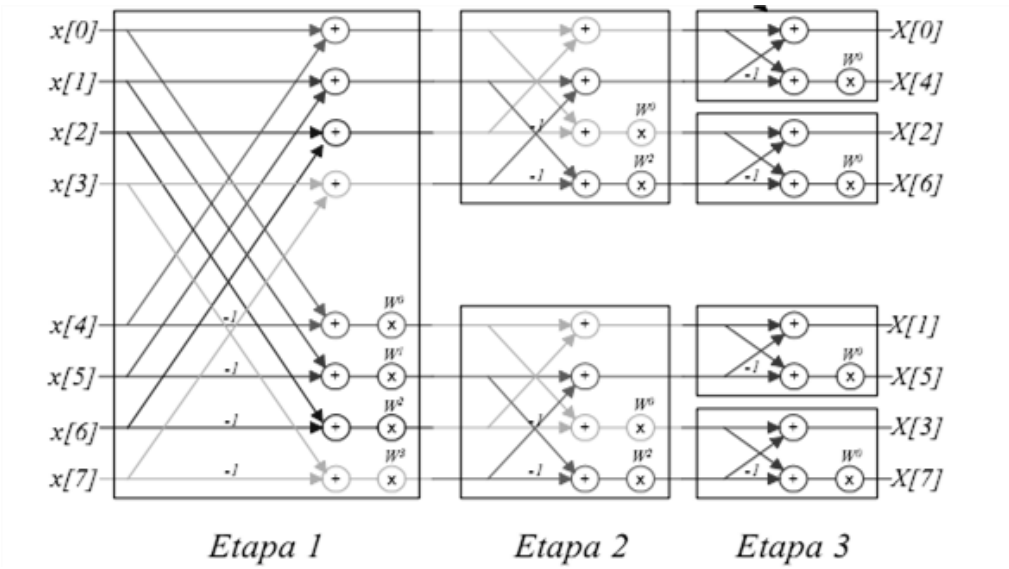


Figura 2.17. *Representación de la secuencia de salida de la FFT (basada en DIF para N=8).*

Intercambio de bits (Bit reversal):

En esta etapa se copian los datos del buffer de entrada al buffer de salida, pero la posición que ocupará cada dato en el arreglo de salida vendrá determinada por una inversión de los bits de las posiciones ocupadas en el arreglo original. Este procedimiento se ejemplifica con mayor claridad en la tabla 2.1, para ocho muestras de salida.

Posición de los datos en el arreglo de entrada antes de aplicar inversión de bit		Posición de los datos en el arreglo de salida después de aplicar inversión de bit	
Decimal	Binario	Binario	Decimal
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Tabla 2.1. Etapa de reversión de bit (bit reversal).

Como se muestra en la tabla anterior, el dato de la posición 0, se copiaría en la posición 0 en el arreglo de salida, el de la posición 1 en la posición 4, el de la posición 2 en la posición 2, hasta completar la salida.

2.3 Estructuras de ficheros compatibles.

Un archivo o fichero informático es un conjunto de bits almacenado en un dispositivo. Por lo general un archivo es identificado por un nombre y la descripción de la carpeta o directorio que lo contiene. Estas estructuras facilitan una manera de organizar los recursos usados para almacenar datos en un sistema informático virtual. En lo que concierne al sistema operativo, un archivo es, en la mayoría de los casos, simplemente un flujo unidimensional de bits, que es tratado por el mismo como una única unidad lógica. Un archivo de datos informático normalmente tiene un tamaño, que generalmente se expresa en bytes; en todos los sistemas operativos modernos, el tamaño puede ser cualquier número entero no negativo de bytes hasta un máximo dependiente del sistema. Depende del software que se ejecuta en la computadora el interpretar esta estructura básica como por ejemplo un programa, un texto, una imagen o un sonido, basándose en su nombre y contenido. La manera en que se agrupan los datos en un archivo depende completamente de la persona que diseñe el mismo. Esto ha conducido a una plétora de estructuras de archivo más o menos estandarizadas para todos los propósitos imaginables, desde los más simples a los más complejos. La mayoría de estos son usados por programas de computadora. Estos

programas crean, modifican y borran datos para su propio uso. Los diseñadores que crean estos sistemas deciden qué archivos necesitan, cómo se van a usar, y (a menudo) sus nombres. En el diseño y posterior implementación de la herramienta de software propuesta en este proyecto, se hace uso de un conjunto de ficheros que de cierta forma contribuyen al intercambio directo de información entre la aplicación desarrollada y el sistema operativo que soporta a la misma, como vía esencial para garantizar la compatibilidad de esta con otros entornos de trabajo. De esta forma una de las tareas esenciales planteadas en las consideraciones previas del diseño fue el hecho de contar con una estructura de trabajo que permitiera desde la aplicación manipular de forma eficiente un conjunto de ficheros y vincularlos al entorno de trabajo diseñado. A continuación se enuncian los principales ficheros compatibles con la aplicación.

- Fichero genérico (.sbio).
- Fichero basado en formato 2-12 (.dat).
- Formato de archivo de audio (.wav).
- Fichero de texto de Windows (.txt).

2.3.1 Fichero genérico (.sbio)

Con el objetivo de garantizar el flujo adecuado de los datos que manipula internamente la aplicación, se diseñó una estructura de archivo binario que entre otras características brinda la posibilidad de adaptar a su formato los parámetros indispensables para el control de una forma de onda simple. El fichero de tipo binario diseñado propone la siguiente estructura (Tabla 2.2):

ORDEN	PARÁMETROS	RESOLUCIÓN (bit)
1	Número de muestra de la señal(N)	64
2	Período de muestreo de la señal(T_s)	64
3	Valor mínimo de tiempo (X_0)	64
4	Valor máximo de tiempo (X_n)	64
5	Valor mínimo de amplitud (Y_0)	64
6	Valor máximo de amplitud (Y_n)	64
7	Trama de datos de la señal (Arreglo[N])	64

Tabla 2.2. *Estructura de datos de los ficheros genéricos (.sbio).*

La extensión de este tipo de fichero es (.sbio), y es precisamente el formato que asume por defecto la aplicación para exportar e importar archivos basados en la estructura anterior.

2.3.2 Fichero basado en formato 2-12 (.dat)

El propósito fundamental de implementar este recurso en la aplicación, está relacionado al hecho de incorporar en la misma una herramienta que de cierta forma posibilite la compatibilidad de esta, con uno de los formatos más usados internacionalmente en la manipulación de bioseñales, el formato de archivo 2-12, siendo este sin dudas, un patrón estándar en lo que a almacenamiento de señales electrocardiográficas se refiere. Facilitando de esta forma el acceso directo al contenido ofrecido por bases de datos de señales que en la mayoría de los casos son evaluadas por criterios de personal calificado. Prestigiosas redes de trabajo como *Physionet* ofrecen este tipo de servicios en sus bancos de datos (*PhysioBank*) ofreciendo una variedad de formatos de acuerdo a las características de las señales que estén contenida, resultando uno de los más comunes, los ficheros basados en el conocido formato 2-12 (*format* 2-12).

La estructura de este formato se fundamenta esencialmente en que cada muestra define un valor de amplitud a partir de una representación en complemento a dos del mismo, a una resolución de 12 bit. La primera muestra se obtiene de los 12 bits menos significativos de la primera pareja de bytes almacenados (byte menos significativo almacenado en la primera posición). La segunda muestra se forma a partir de los 4 bits restantes de la primera pareja de bytes mencionada anteriormente (serán estos los 4 bits superiores de la segunda muestra de 12-bits) y el byte siguiente (que contiene los 8 bits restantes de la segunda muestra). El proceso se repite para cada par sucesivo de muestras (ver figura 2.18). Cabe destacar que la mayoría de los archivos de señales en *PhysioBank* se escriben en formato 212, el cual se apoya por lo general de otras estructuras con el fin de garantizar una mejor manipulación de sus elementos, entre estas se encuentra el fichero de cabeceras (.hea) en cuyo contenido están presente datos esenciales de su señal correspondiente como su frecuencia de muestreo, número de datos en la trama principal, número de canales, entre otros. En el proceso de implementación se trabajaron con señales descargadas desde *PhysioBank* procedentes de la MIT-DB (*Massachusetts Institute of Technology -Data Base*).

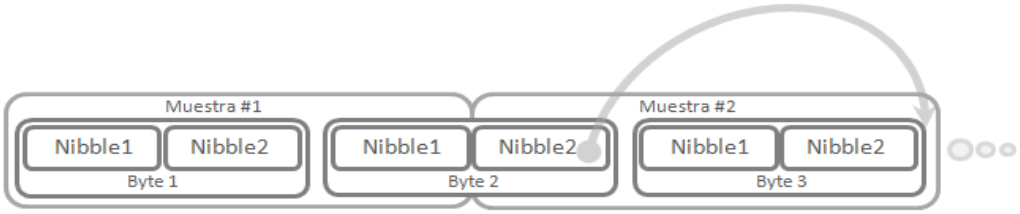


***Figura 2.18**. Estructura de formato 212.*

Para realizar la lectura de ficheros basados en el formato 212, a partir de lo mencionado anteriormente, se utilizó la siguiente lógica:

```
M1 = ((B2 & 0x0F) << 8) | B1                    [2.21]

M2 = ((B2 & 0xF0) << 4) | B3                    [2.22]
```

Donde:

M1: Representa el valor obtenido en la lectura de la primera muestra.

M2: Representa el valor obtenido en la lectura de la segunda muestra.

B1: Representa el valor del byte de la primera posición.

B2: Representa el valor del byte de la segunda posición.

B3: Representa el valor del byte de la tercera posición.

De esta forma, basado en las operaciones aritméticas a nivel de bits relacionados anteriormente, se implementó el algoritmo propuesto.

2.3.3 Formato de archivo de audio (.wav)

WAV (o WAVE), apócope de Waveform Audio File Format, es un formato de audio digital normalmente sin compresión de datos desarrollado y propiedad de Microsoft y de IBM que se utiliza para almacenar sonidos en la PC, admite archivos mono y estéreo a diversas resoluciones y velocidades de muestreo. Su extensión es ".wav". Es una variante del formato RIFF (*Resource Interchange File Format*, formato de fichero para intercambio de recursos), método para almacenamiento en "paquetes", y relativamente parecido al IFF (*Interchange File Format*) y al formato AIFF (*Audio Interchange File Format*) usado por Macintosh [15]. Aunque el formato WAV puede soportar casi cualquier códec de audio. Una de sus grandes limitaciones es que solo se puede grabar un archivo de hasta 4

gigabytes, que equivale aproximadamente a 6,6 horas en calidad disco compacto. Esta es una limitación propia del formato, independientemente de que el sistema operativo donde se utilice sea Windows u otro distinto, y se debe a que en la cabecera del fichero se indica la longitud del mismo con un número entero de 32 bit, lo que limita el tamaño del mismo a 4 GB. A pesar de sus limitaciones continúa jugando un papel protagónico en diferentes esferas, y es precisamente una de estas la manipulación mediante software de formas de ondas hechas a la medida desde aplicaciones específicas que de cierta manera tengan la capacidad de manipular este formato de sonido. Posibilitando así la compatibilidad con diversos entornos de trabajo. A continuación se plantea la estructura interna del formato .WAV. La estructura global de un archivo WAV la podemos dividir en tres partes (ver tabla 2.3):

Nombre	Tamaño (bytes)	Descripción
rID	4	Cadena 'RIFF' para identificar el fichero.
rLen	4	Tamaño del sector rData.
rData	rLen	Datos de formato y audio.

Tabla 2.3. Estructura global de un archivo .wav.

A su vez, "rData" se divide en tres partes (tabla 2.4):

Nombre	Tamaño (bytes)	Descripción
wID	4	Cadena 'WAVE' para identificar el fichero WAV.
fData	24	Estructura con los parámetros del formato utilizado en la codificación del audio.
wData	¿?	Datos del audio.

Tabla 2.4. Estructura "rData" de un archivo .wav.

Por otro lado, la estructura de 'fData' está compuesta por (tabla 2.5):

Nombre	Tamaño (bytes)	Descripción
fID	4	Cadena 'fmt ' para identificar el formato.
fLen	4	Tamaño del bloque de formato.
wFormatTag	2	Formato utilizado (generalmente es 1=PCM).
nChannels	2	Número de canales (1=mono, 2=estéreo).
nSamplesPerSec	4	Frecuencia de muestreo (en Hz).
nAvgBytesPerSec	4	nChannels*nSamplesPerSec*(nBitsPerSample/8) Para estimar el tamaño requerido en el búfer.
nBlockAlign	2	nChannels*(nBitsPerSample/8) Para la alineación del búfer.
FormatSpecific	2	Medida del muestreo, en bits (8 o 16).

Tabla 2.5. Estructura "fData" de un archivo .wav.

Y, por último, 'wData' se estructura de la siguiente forma (Tabla 2.6):

Nombre	Tamaño (bytes)	Descripción
dID	4	Identificación de datos.
dLen	4	Longitud del bloque 'dData'.
dData	dLen	Datos de las muestras del audio.

Tabla 2.6. Estructura "wData" de un archivo .wav.

Con el control de la manipulación de este tipo de ficheros la aplicación de software propuesta se dota de una poderosa herramienta que permitirá facilitar el intercambio de información desde esta hacia otros entornos de trabajo profesionales como Proteus®, facilitando en este la inyección de señales en circuitos simulados desde este ambiente, así como desde otros espacios como MATLAB® por medio del cual se puede acceder a la información contenida en un fichero basado en formato ".WAV" con gran facilidad. Así como facilitar la generación de tonos desde el propio programa, además de posibilitar otras aplicaciones propias del presente formato.

2.3.4 Fichero de texto de Windows (.txt)

Los ficheros de texto de Windows cuya extensión (.txt) es una de las más comunes en ordenadores personales, es una de las más usadas por los sistemas diseñados para la generación y procesamiento de señales, debido a que sobre el mismo resulta muy sencillo realizar procesos tanto de lectura como de escritura alterando solo el contenido necesario en la información manipulada. La aplicación diseñada en la presente investigación tiene la capacidad de exportar e importar ficheros en formato ".txt", generando una trama de datos de salida en forma de vector columna en correspondencia a cada una de las muestras de la señal estudiada, cuya resolución se presenta a partir de elementos reales de doble precisión con un alcance 64 bit. De igual forma ocurre al momento de importar un fichero de texto basado en elementos reales en forma de vector columna.

3. CAPÍTULO 3. EVALUACIÓN DE RESULTADOS

A partir de la implementación y posterior puesta en funcionamiento de la herramienta de software descrita en el presente proyecto, se puede realizar un análisis integral de la misma.

Partiendo de las características individuales de cada uno de los algoritmos propuestos y de forma general del diseño integral de esta, en el ***anexo #1*** se muestra el diagrama de flujo general de la herramienta de software "GESEBIOv1.0" propuesta como resultado final del presente proyecto de tesis.

3.1 Herramienta de software "GESEBIOv1.0"

GESEBIOv1.0 es una herramienta de software diseñada principalmente con el objetivo de realizar tareas tanto de síntesis como de procesamiento de señales, ofreciendo para ello un conjunto de herramientas implementadas con tal fin. A partir del entorno de trabajo propuesto por esta aplicación, el usuario tiene la posibilidad de interactuar de forma directa e intuitiva con una interfaz gráfica, capaz de dar solución a un conjunto de necesidades que en materia de generación y procesamiento de señales pudieran ser necesarias. A continuación se citan las principales opciones que la herramienta es capaz de manipular.

- **Posibilidad de generar formas de ondas con las siguientes características:**
  - Forma de onda de señal de ECG.
  - Forma de onda de señal de PPG.
  - Forma de onda de señal cuadrada.
  - Forma de onda de señal triangular.
  - Forma de onda de señal diente de sierra.
  - Forma de onda de señal trapezoidal.
  - Forma de onda de señal sinusoidal.

- **La aplicación posibilita el dibujo manual y la edición de señales a partir de:**
  - La modificación directa de la morfología de formas de ondas mediante el *mouse*.

- La selección y eliminación de tramos de señales.
- La simulación del efecto de la cuantificación sobre una señal digital.

- **Filtrado digital de señales mediante las siguientes técnicas:**
  - Filtros digitales de tipo FIR.
  - Filtros digitales de tipo IIR.
- **Adición de ruido o interferencia:**
  - Incorporar ruido blanco gaussiano.
  - Incorporar interferencia de la red de alimentación eléctrica.
- **Manipulación de la tarjeta de sonido de la PC:**
  - Permite la generación de tonos por la tarjeta de sonido.
  - Permite la captura de sonidos por micrófonos.
- **Compatibilidad:**
  - Permite importar y exportar señales en formato de audio ".wav"**.**
  - Permite importar y exportar señales en formato de texto de Windows ".txt"**.**
  - Permite importar y exportar señales en formato genérico (propio de la aplicación)".sbio"**.**
  - Permite importar señales en formato 212 (.dat**)** y ficheros asociados (.hea) para señales específicas**.**
- **Operaciones para el análisis de señales:**
  - Obtención del espectro en frecuencia de una señal a partir del cálculo de la FFT.
  - Cálculo de la IFFT.
- **Otras opciones:**
  - Configuración y chequeo de parámetros de la señal (período de muestreo, número de muestras).
  - Exportar e importar comentarios asociados a señales específicas.
  - Exportar señal en formato de imagen desde el área de trabajo.
  - Exportar espectro en frecuencia de una señal en formato ".txt".

- Ayuda preliminar a partir de documentación complementaria, con la descripción de las principales funcionalidades de la aplicación.

Cada una de las funcionalidades citadas anteriormente se integran dentro de un ambiente de trabajo diseñado con el fin de proporcionar al usuario acceso rápido y sencillo a cada una de las mismas. La aplicación de forma general está estructurada en tres pantallas, desde las cuales se ejecutan diferentes procedimientos individuales que de cierta forma gestionan el correcto funcionamiento de todo el entorno de diseño propuesto. Estas pantallas son:

- Pantalla de inicialización.
- Pantalla de entrada.
- Espacio de trabajo.

3.1.1 Pantalla de inicialización

Desde la pantalla de inicialización(ver figura 3.1) la aplicación tiene la capacidad de ejecutar un conjunto de instrucciones encaminadas a inicializar los parámetros de las tablas que almacenan las muestras correspondientes a las formas de onda de las señales tanto de ECG como de PPG, almacenadas en los ficheros ECG_01_1000.dat y FOTO_01_1000.dat respectivamente.

Figura 3.1. *Pantalla de inicialización de ¨GESEBIOv1.0¨*

Este procedimiento lo ejecuta la aplicación haciendo un chequeo previo de la existencia en el directorio de la misma, de los archivos anteriormente mencionados (ECG_01_1000.dat y FOTO_01_1000.dat), para posteriormente realizar la lectura de estos, con el objetivo de

hacerlos visibles al resto del entorno de trabajo diseñado. El vínculo de esta pantalla con el resto de la aplicación se ejecuta de forma automática en un tiempo relativamente corto (aproximadamente 10 segundos) cuya exactitud depende directamente de los recursos computacionales con que disponga la PC donde se ejecute el software.

3.1.2 Pantalla de entrada.

La pantalla de entrada (ver figura 3.2) es la que sigue a pantalla de inicialización, desde esta el usuario puede acceder a las "Opciones de entrada" que ofrece la herramienta.

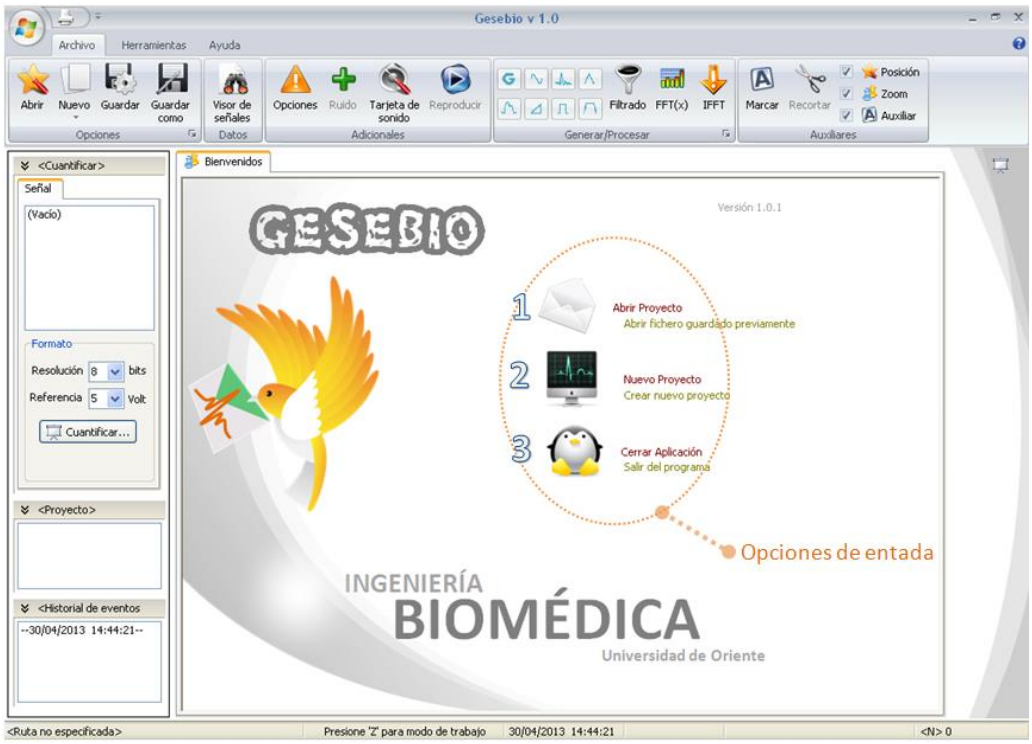


Figura 3.2. *Pantalla de estrada de ¨GESEBIOv1.0¨*

Las "opciones de entrada" que ofrece esta ventana le permiten al usuario abrir desde la propia vista, un proyecto existente de extensión ".sbio" (Opción #1), así como también se puede acceder a la próxima Pantalla (Espacio de trabajo) (Opción #2) si lo que se desea es crear un nuevo proyecto, o simplemente el usuario puede cerrar la aplicación (Opción #3).

3.1.3 Espacio de trabajo.

El área correspondiente al espacio de trabajo (ver figura 3.3) constituye la pantalla principal de la aplicación en la cual la herramienta integra un conjunto de instrucciones encaminadas a optimizar los recursos necesarios para la manipulación de un proyecto determinado.

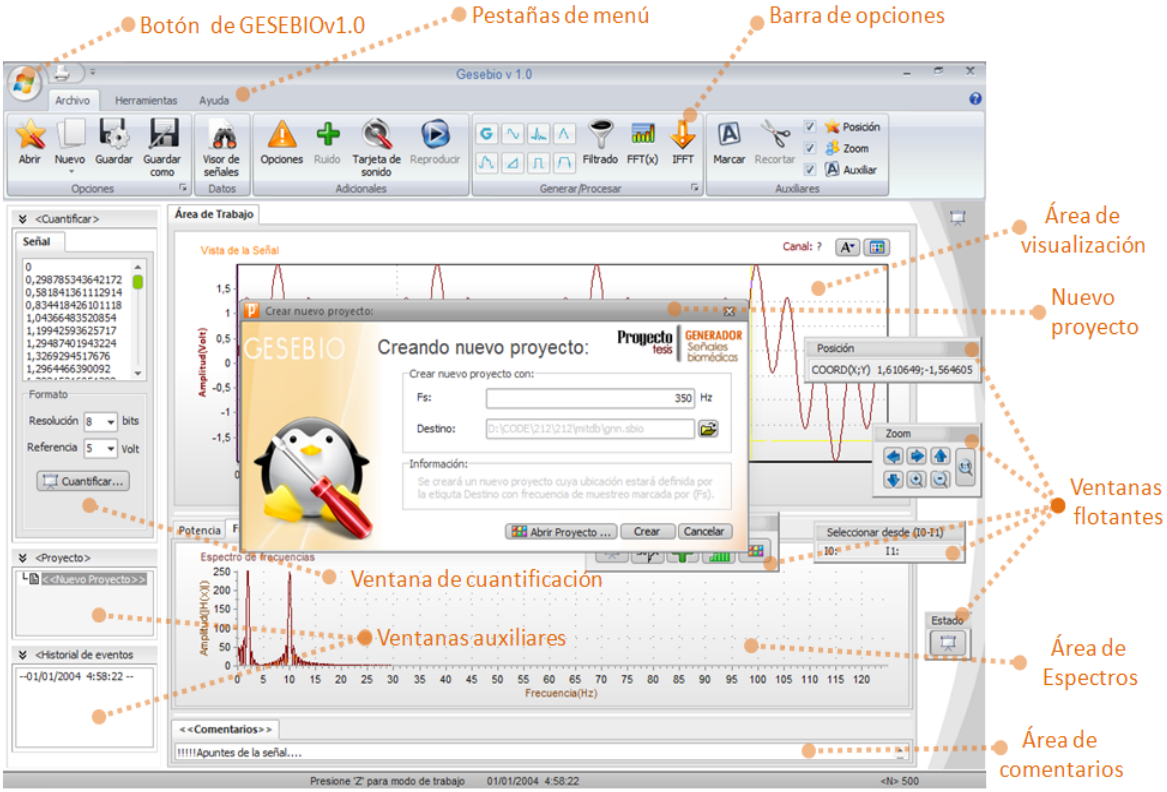


Figura 3.3. *Espacio de trabajo de ¨GESEBIOv1.0¨*

Desde esta ventana se puede acceder a opciones disponibles en esta como:

(a) Ventana de configuración previa para crear nuevo proyecto.

(b) Botón de GESEBIOv1.0.

(c) Barra de opciones.

(d) Espacio de trabajo.

- Área de visualización.
- Ventanas flotantes, área de espectros y área de comentarios.
- Ventana de cuantificación.

- Ventanas auxiliares.

Estas opciones son explicadas en detalle en la documentación complementaria, anexada a la herramienta de software GESEBIOv1.0.

3.2 Pruebas realizadas a la aplicación

Con el objetivo de verificar el correcto funcionamiento de cada uno de los algoritmos implementados en la herramienta de software propuesta, se procedió a realizar algunas pruebas prácticas, en la que se incluyeron las opciones fundamentales de la aplicación.

Prueba #1. Para el desarrollo de esta prueba se describieron los siguientes pasos:

1. Generación de una señal sintética de ECG.

   Generando señal de ECG con las siguientes características.

   La señal de ECG diseñada desde la ventana de generación de la aplicación tiene las siguientes características (ver figura 3.4).

Parámetros	Latido/minuto	Repetir(veces)	Amplitud	Offset	Frecuencia de muestreo
Valor	60	10	0,001 V	0 V	1000 Hz

***Tabla 3.1**. Parámetros de señal de ECG generada desde GESEBIOv1.0.*

2. Determinación del espectro en frecuencia de la señal generada a partir del cálculo de la FFT para 32768 puntos (Ver figura 3.5).
3. Adición de interferencia procedente de la red de 60 Hz a la señal de ECG generada anteriormente (Ver figura 3.6).
4. Determinación del espectro en frecuencia de la señal con interferencia obtenida a partir del paso anterior. Con el objetivo de establecer una comparación con el espectro obtenido en el paso 2, y ver la influencia de la interferencia asociada (Ver figura 3.7).
5. Aplicar un filtro Notch con una frecuencia central de 60 Hz y un ancho de banda de 10 (Ver figura 3.8).
6. Aplicar FFT a la señal resultante (Ver figura 3.9).

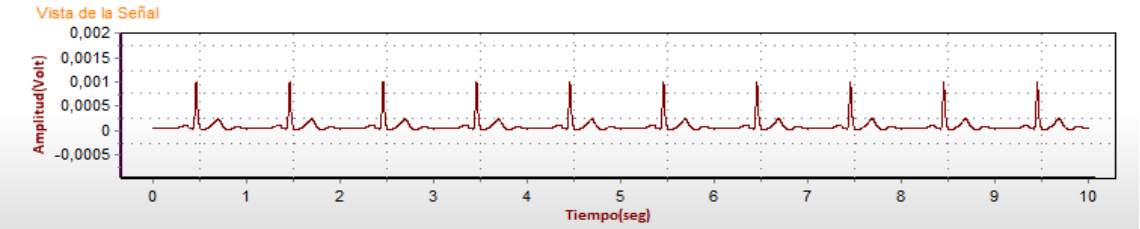


***Figura 3.4**. Paso #1. Señal de ECG generada desde GESEBIOv1.0.*

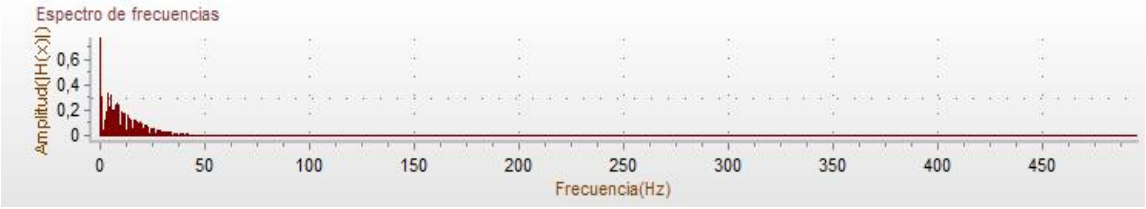


***Figura 3.5**. Paso #2. Espectro de frecuencias de la señal de ECG generada (Amplitud-frecuencia).*

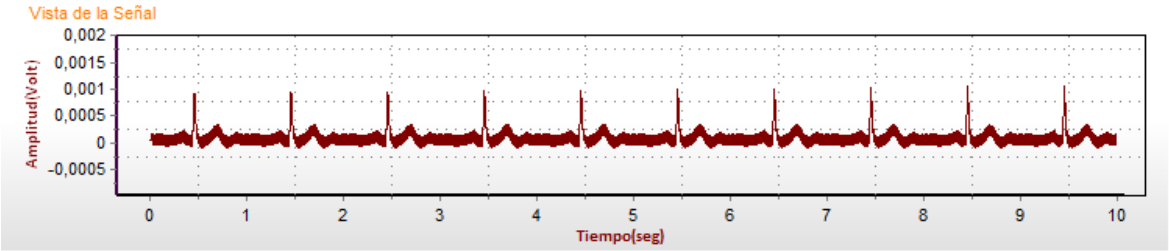


***Figura 3.6**. Paso #3. Señal de ECG con interferencia de la red (60 Hz).*

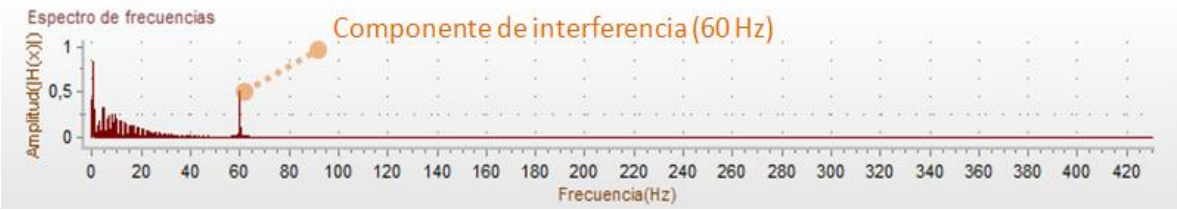


***Figura 3.7**. Paso #4. Espectro de frecuencias de la señal con interferencia (Amplitud-frecuencia).*

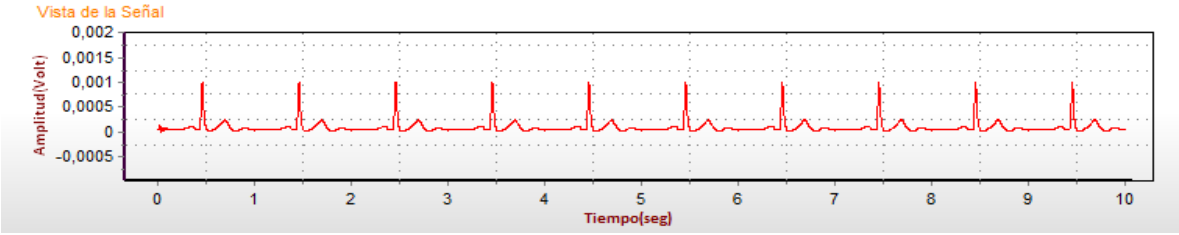


***Figura 3.8.** Paso #5. Señal del paso #3 filtrada con filtro Notch.*

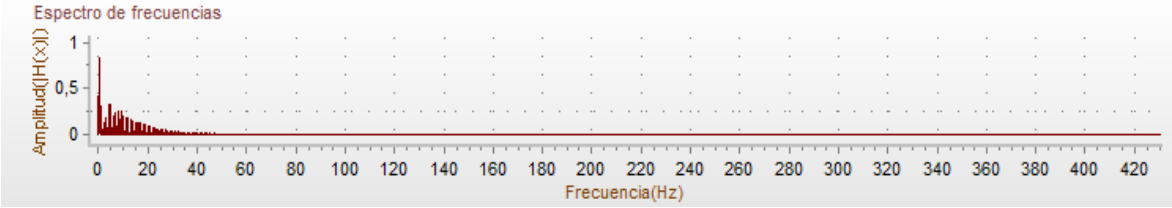


***Figura 3.9**. Paso #6. Espectro de frecuencias de la señal filtrada con filtro Notch.*

De esta forma se pudo verificar de forma cualitativa el correcto funcionamiento de cada uno de los algoritmos implementados, presentes en la prueba anterior. Similar a esta, se realizaron otras comprobaciones con el resto de las señales propuestas por el generador de GESEBIOv1.0, incluyendo además otros tipos de filtros y añadiendo ruido blanco gaussiano, obteniéndose en cada una de estas, resultados satisfactorios.

Prueba #2. En esta prueba se adquirió una señal de EGC contenida en un fichero ".dat" basado en formato 212, asociado a este fichero se encuentra su respectivo archivo cabecera (.hea), en el cual está toda la información de la señal en cuestión (frecuencia de muestreo, número de muestras y canales, nombre de la señal entre otros parámetros), con el objetivo de emplear esta señal en una simulación desde VSM Proteus®. Estos ficheros fueron tomados de la MIT-DB. Una vez obtenida la señal (Ver figura 3.10) se procede a ejecutar los siguientes pasos:

- Se seleccionó para el trabajo, solo un tramo específico (aproximadamente 10 segundos) de la señal adquirida, modificándose de forma manual la morfología del tramo comprendido por los primeros 5 segundos de esta. Añadiendo además, ruido blanco gaussiano de amplitud 0,0001 V, al tramo comprendido a partir de los 5 segundos en adelante. (ver figura 3.11).
- Se exportó el tramo de señal obtenido, en formato *.wav* para de esta forma emplearlo desde la herramienta de simulación Profesional VSM Proteus®. (Ver figura 3.12).

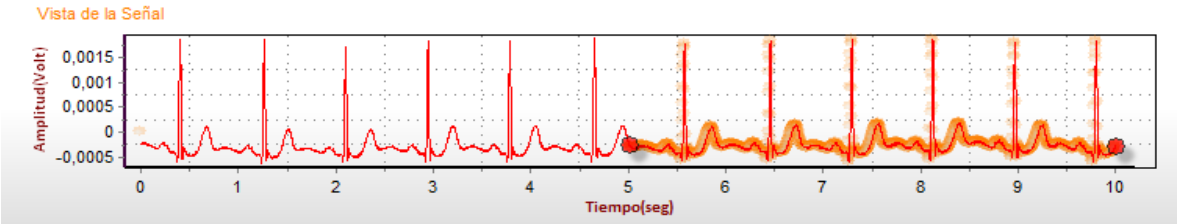


***Figura 3.10**. Selección de un tramo de una señal de ECG de formato 2-12 adquirida desde la aplicación.*

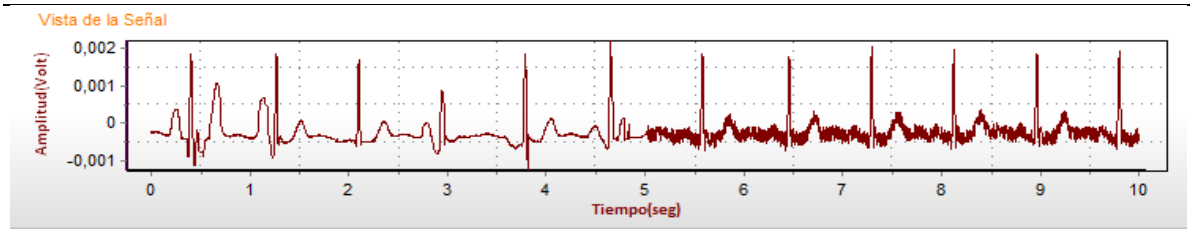


***Figura 3.11**. Señal manipulada de forma manual (0-5 segundos) y con ruido blanco aleatorio asociado (5-10 segundos).*

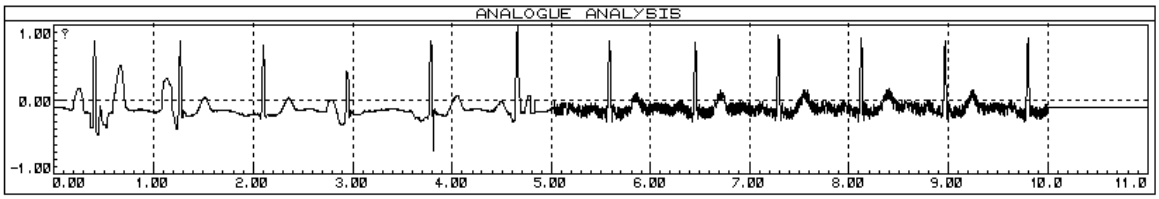


***Figura 3.12**. Señal exportada en formato ".wav" manipulada desde VSM Proteus®.*

De esta forma se verificó el correcto funcionamiento de cada uno de los algoritmos que intervinieron en la prueba anterior.

Prueba #3. La prueba consiste en generar una señal sintética de PPG a una frecuencia de muestro de 250 Hz. Y simular el efecto que ejerce la cuantificación en una señal digital para varios niveles de resolución binaria y para valores típico de referencia (rango dinámico) de conversores analógico-digitales prácticos, (asumiendo la señal generada como una variable de origen analógico). Para la presente prueba se realizaron los siguientes pasos:

- Se generó una señal sintética de PPG, introduciendo variabilidad en la ocurrencia temporal de cada pulso, generando 10 períodos de la forma de onda, a partir de valores enteros generados aleatoriamente en un rango de 40-80 latidos por minuto y una amplitud máxima de 3 V (Ver figura 3.13 y 3.14) .

- Cuantificación de la señal generada en el paso anterior a una resolución de 8 bit y una referencia de 3V (Ver figura 3.15, tomada desde el visor de señales de GESEBIOv1.0).

- Cuantificación de la señal generada en el primer paso a una resolución de 4 bit y una referencia de 3V (Ver figura 3.16, tomada desde el visor de señales).

- Cuantificación de la señal generada en el primer paso a una resolución de 8 bit y una referencia de 2.5V (Ver figura 3.17, tomada desde el visor de señales).

***Figura 3.13**. Parámetros de señal de PPG generada desde GESEBIOv1.0.*

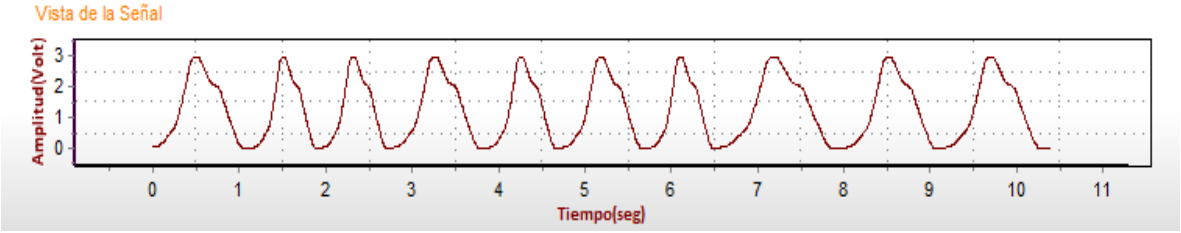


***Figura 3.14**. Señal de PPG generada desde GESEBIOv1.0.*

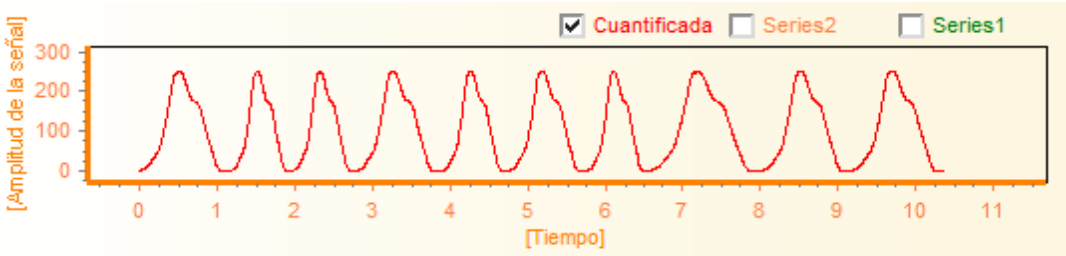


***Figura 3.15**. Señal de PPG cuantificada a 8 bit de resolución y una referencia de 3V.*

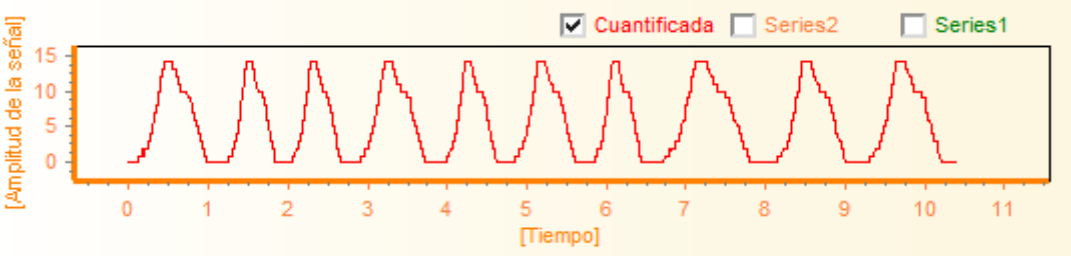


***Figura 3.16**. Señal de PPG cuantificada a 4 bit de resolución y una referencia de 3V.*

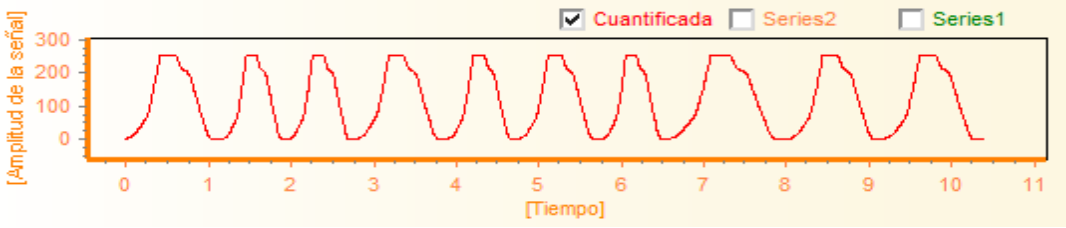


***Figura 3.17**. Señal de PPG cuantificada a 8 bit de resolución y una referencia de 2.5V.*

De esta forma se pudo verificar el correcto funcionamiento de cada una de las instrucciones que intervienen en la presente prueba.

Prueba #4. La presente prueba se dividirá en dos etapas, en la primera se realizará la captura de una señal de voz desde la tarjeta de sonido de la PC. Y en la segunda etapa se generará una señal sinusoidal de frecuencia igual a 1 kHz, con el objetivo de reproducir el tono por la tarjeta de sonido. Para ello se siguió la siguiente metodología.

- Configurar la tarjeta de sonido de la PC desde la aplicación, para una frecuencia de muestreo de 1600 Hz y 2110 muestras a 8 bit de resolución, para proceder a realizar la captura de la señal procedente desde micrófono (Ver figura 3.18).

- Generación de una señal sinusoidal de 1V de amplitud, 1kHz de frecuencia, 0º de fase, 0V de offset y 1 segundo de duración a una frecuencia de muestreo de 10 kHz (Ver figura 3.19). Para posteriormente exportarla en formato de audio y generar el tono en cuestión por la tarjeta de sonido, con las correspondientes opciones de GESEBIOv1.0.

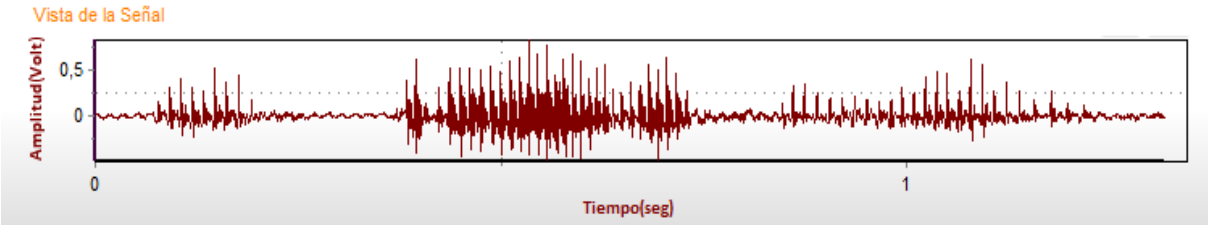


***Figura 3.18**. Señal de audio adquirida desde la tarjeta de sonido de la PC.*

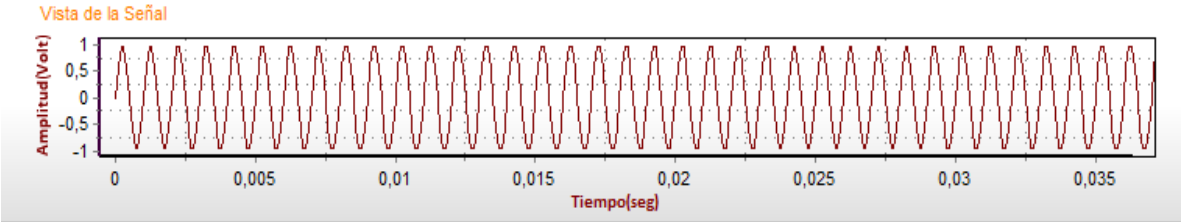


***Figura 3.19**. Señal sinusoidal generada a 1 kHz.*

En la presente prueba se verificó el correcto funcionamiento de los métodos implementados, tanto para garantizar la captura, como la generación de tonos por la tarjeta de sonido de la PC. En la primera etapa de la prueba se obtuvo la señal de voz esperada, la cual fue exportada en formato de audio para verificar su contenido tal y como se muestra en la figura 3.18. En la segunda etapa se verificó el tono generado a partir de un *script* implementado desde MATLAB®, en el cual se obtuvieron resultados idénticos.

3.3 Análisis de resultados de algoritmos implementados

Con el objetivo de validar los resultados de los métodos encaminados a garantizar el correcto análisis y procesamiento de las señales manipuladas por la aplicación, se procedió realizar una comparación de las respuestas obtenidas por los algoritmos implementados para GESEBIOv1.0 y las obtenidas por la herramienta profesional MATLAB R2010a®, los algoritmos sometidos a prueba fueron los siguientes:

- Algoritmo para el cálculo de la FFT.
- Algoritmo para filtros digitales de tipo FIR.
- Algoritmo para filtros digitales de tipo IIR.

Evaluación de algoritmo para el cálculo de la FFT.

Para realizar la comparación de los resultados del cálculo de la FFT, obtenidos a partir de la herramienta de software GESEBIOv1.0, con datos obtenidos por la función fft(X,n) de MATLAB R2010a®, se procedió a realizar los siguientes pasos:

- Obtención de un espectro de 8 muestras, desde GESEVIOv1.0, a partir de una señal llamada S1 generada desde esta aplicación.

- Exportar señal S1 desde GESEBIOv1.0 en formato de texto para facilitar su uso desde MATLAB®.
- Obtención de un espectro de 8 muestras desde MATLAB®, a partir de la señal S1.
- Determinar del error relativo porcentual entre cada una de las muestras del espectro obtenido desde GESEBIOv1.0 y MATLAB® a partir de la expresión 3.1. Para determinar posteriormente el error medio introducido por cada muestra (ver tabla 3.2).

$$E_r = \frac{Xe - Xo}{Xo} * 100 \qquad 3.1$$

Donde: Xe: Representa el valor esperado (valores del Espectro de MATLAB®).

Xo: Representa el valor obtenido (valores del Espectro de GESEBIOv1.0).

Para la obtención del espectro desde MATLAB® se implementó un *script* que determina el espectro de amplitud de la señal analizada para una FFT de 16 puntos. En esta solo se analizará la primera mitad del espectro ya que el resto de los valores son idénticos solo que están invertidos.

No	Espectro de MATLAB®	Espectro de GESEBIOv1.0	Error relativo (%)
0	12.6054	12.605371	0,00023006
1	4.8531	4.853073	0,00055635
2	0.7984	0.798372	0,00350714
3	0.3421	0.342065	0,01023197
4	0.1968	0.196834	0,01727344
5	0.1336	0.133568	0,02395783
6	0.1023	0.102308	0,00781953
7	0.0873	0.087254	0,05271965
		Error promedio(%)	0,01383997

Tabla 3.2. *Espectros obtenidos en GESEBIOv1.0 y MATLAB® (para una FFT de 16 puntos).*

Como se puede apreciar en la tabla anterior el error obtenido fue de 0,01383997% garantizando la eficiencia del algoritmo propuesto. De esta forma se implementaron otras pruebas similares con números de puntos para la FFT diferentes, (64, 128, 512, 1024). Obteniéndose resultados similares al anterior.

Evaluación de algoritmo para filtros digitales de tipo FIR implementados desde GESEBIOv1.0.

Para la implementación de esta prueba se hizo uso de la herramienta *fdatool* que proporciona MATLAB®, a través de la cual se obtuvieron los coeficientes de los filtros que posteriormente fueron comparados con los diseñados desde la aplicación GESEBIOv1.0. Estos filtros fueron diseñados por el método de ventanas que es precisamente el que propone el software GESEBIOv1.0. En la tabla 3.3 se muestran el valor medio del error relativo porcentual incorporado por cada tipo de filtro de manera individual. Este error es obtenido promediando el error aportado entre cada uno de los coeficientes, calculados desde *fdatool* y GESEBIOv1.0. El orden escogido para la prueba es 10, empleando la ventana de Hamming, una frecuencia de muestreo de 250 y con las siguientes frecuencias de corte para cada tipo de filtro:

Paso bajo: 50 Hz.

Paso alto: 50 Hz.

Paso banda: 50-100 Hz.

Supresor de banda: 50-100 Hz.

Tipo de filtro	Paso bajo	Paso alto	Paso banda	Supresor de banda
Error (%)	0.201	0.051	0	0,414

***Tabla 3.3**. Error relativo incorporado por filtros FIR de onden 10.*

De esta forma se puede verificar que los errores incorporados son mínimos. Se ejecutaron además pruebas similares para filtros FIR con el resto de las ventanas disponible desde GESEBIOv1.0 y ordenes diferentes, obteniéndose valores similares.

Evaluación de algoritmo para filtros digitales de tipo IIR implementados desde GESEBIOv1.0.

Para el caso de los filtros IIR se procedió de forma similar a la prueba anterior. El orden escogido para la prueba es 2, una frecuencia de muestreo de 250 y con las siguientes frecuencias de corte para cada tipo de filtro:

Paso bajo: 50 Hz.

Paso alto: 50 Hz.

Paso banda: 50-100 Hz.

Supresor de banda: 50-100 Hz.

Notch: frecuencia central de 250 con un ancho de banda de 5.

Tipo de filtro	Paso bajo	Paso alto	Paso banda	Supresor de banda	Notch
Error (%)	0	0	0.0610	0.0610	0.0158

Tabla 3.4. *Error relativo incorporado por filtros IIR de onden 2.*

Los principales problemas están asociados, a las aproximaciones empleadas para el cálculo de los coeficientes de cada filtro, particularmente en las ponderaciones de ganancias asumidas para cada una de las secciones de bicuadráticas relacionadas con los filtros paso banda y supresor de banda específicamente. De esta forma se recomienda mejorar en posteriores trabajos, los resultados en la implementación de estos.

CONCLUSIONES

En el presente trabajo:

- Se describieron los fundamentos teóricos que sustentan a los principales algoritmos implementados en la herramienta de software propuesta.
- Se implementaron algoritmos para el cómputo de la FFT, el diseño de filtros FIR por el método de ventanas, y de filtros IIR a partir del método de transformación bilineal empleado para el cálculo de las aproximaciones de Butterworth. Así como otros encaminados a garantizar la síntesis y edición de señales desde la aplicación diseñada.
- Propuesta de la herramienta de software GESEBIOv1.0.
- Se realizó un análisis en cuanto al desempeño de cada uno de los algoritmos implementados, obteniéndose los resultados esperados.

Por lo antes expuesto se confirma la hipótesis planteada, al disponer de una herramienta de software para la síntesis de Bioseñales, a partir de la cual se pueden diseñar patrones de forma de ondas adaptadas a las necesidades del usuario. Dando la posibilidad de integrarla a la docencia como una nueva herramienta de trabajo.

RECOMENDACIONES

Con el objetivo de mejorar el rendimiento y ampliar las prestaciones de la herramienta implementada se proponen las siguientes recomendaciones.

- Optimizar los algoritmos de cálculo empleados en el procesamiento y edición de señales.
- Incorporar nuevos método de filtrado de señales.
- Añadir nuevas técnicas para la generación de nuevas formas de onda por software.
- Asegurar la compatibilidad de nuevos formatos de archivos, que posibiliten el intercambio de información con la aplicación.

VALORACIÓN ECONÓMICA

En el mercado internacional los precios de las herramientas de software con prestaciones similares a la propuesta en el presente proyecto, oscilan alrededor de los 600 USD. Por lo que resulta relevante el hecho de contar con una aplicación con tales características y disponer además de un paquete de herramientas propias, implementadas desde un lenguaje de programación estándar como C++. Este proyecto facilitará el uso de una herramienta práctica en asignaturas vinculadas al tratamiento y procesamiento de señales.

El costo de los materiales y servicios para el diseño e implementación del software propuesto, es de aproximadamente $3200 MN. Para el cálculo de este valor se tuvieron en cuenta el consumo eléctrico de los medios de cómputo, el gasto de salario y herramientas de software empleadas.

Puede anticiparse que la sinergia entre el valor agregado en las prestaciones (know how) y el bajo costo de producción, sumado esto a la experiencia acumulada en el presente trabajo, permite prever que la aplicación pudiera ser competitiva en el mercado internacional.

REFERENCIAS BIBLIOGRÁFICAS

[1] [Colectivo de Autores] "Electronic test instruments". BK Precisión®. Diciembre 2010.

[2] Sánchez, Daniel. "Procesado y transmisión de señales Biomédicas para el diagnóstico de Trastornos y enfermedades del sueño", Universidad de Cádiz, Febrero 2008.

[3] Barea Navarro,R. "Introducción y Conceptos Básicos de la Instrumentación. Biomédica", Departamento electrónica. Universidad Alcalá, Agosto 2002.

[4] Bronzino, J.D. "The Biomedical Engineering Handbook", 2da edición, 2000.

[5] Barea Navarro, Rafael. "Sistemas de Acondicionamiento y Adquisición de Señales Bioeléctricas". Departamento electrónica. Universidad Alcalá, Noviembre 2002.

[6] Narváez Raúl. Jaramillo Andrés. "Diferenciación entre electrocardiogramas normales y arrítmicos usando análisis en frecuencia". Universidad del Rosario, Colombia. 2004.

[7] Salazar Guillermo, Franco. "Electrocardiografía." Capitulo #3. 5ta edición, 2005.

[8] Guyton & Hall. "Tratado de fisiología médica", 12ma edición, España. 2011.

[9] R. Rubert, M. Geordanis. "Herramienta para el análisis de la Variabilidad de la Frecuencia Cardiaca en registros ECG de larga duración". Universidad de Oriente. Santiago de Cuba, 2007.

[10] López Silva, S. M. "Utilidad de la fotopletismografía por transmisión con diodos láser infrarrojos en el estudio de la perfusión visceral", vol. 38, núm. 1, 2005.

[11] Bernal Jiménez, Andrés. "Desarrollo de un software para la medición del estrés" Centro de investigación y de estudios avanzados del instituto politécnico nacional, Unidad Zacatenco, México, D.F. Abril 2010.

[12] Gómez, Emilia. "Síntesis aditiva", Escuela Superior de Música de Catalunya, septiembre de 2009.

[13] Coombs, Clyde F. "Electronic Instrumentation Handbook". Mc Graw, Hill. Inc. 2da edición. 2005.

[14] Guerrero Martínez, Juan F. "Procesado Digital de Bioseñales", Escola Técnica Superior d' Enginyeria, Universitat d' Valéncia, Departament d´ Enginyeria Electrónica, 2011.

[15] Jiménez De Parga Bernal, Carlos. "Desarrollo de aplicaciones de audio en C++: un enfoque práctico". Escuela técnica superior de ingeniería informática. España. Diciembre 2010.

[16] Posadas Yagüe, Juan Luis. "Transformada Rápida de Fourier e Interpolación en Tiempo Real". Universidad Politécnica de Valencia. 2000.

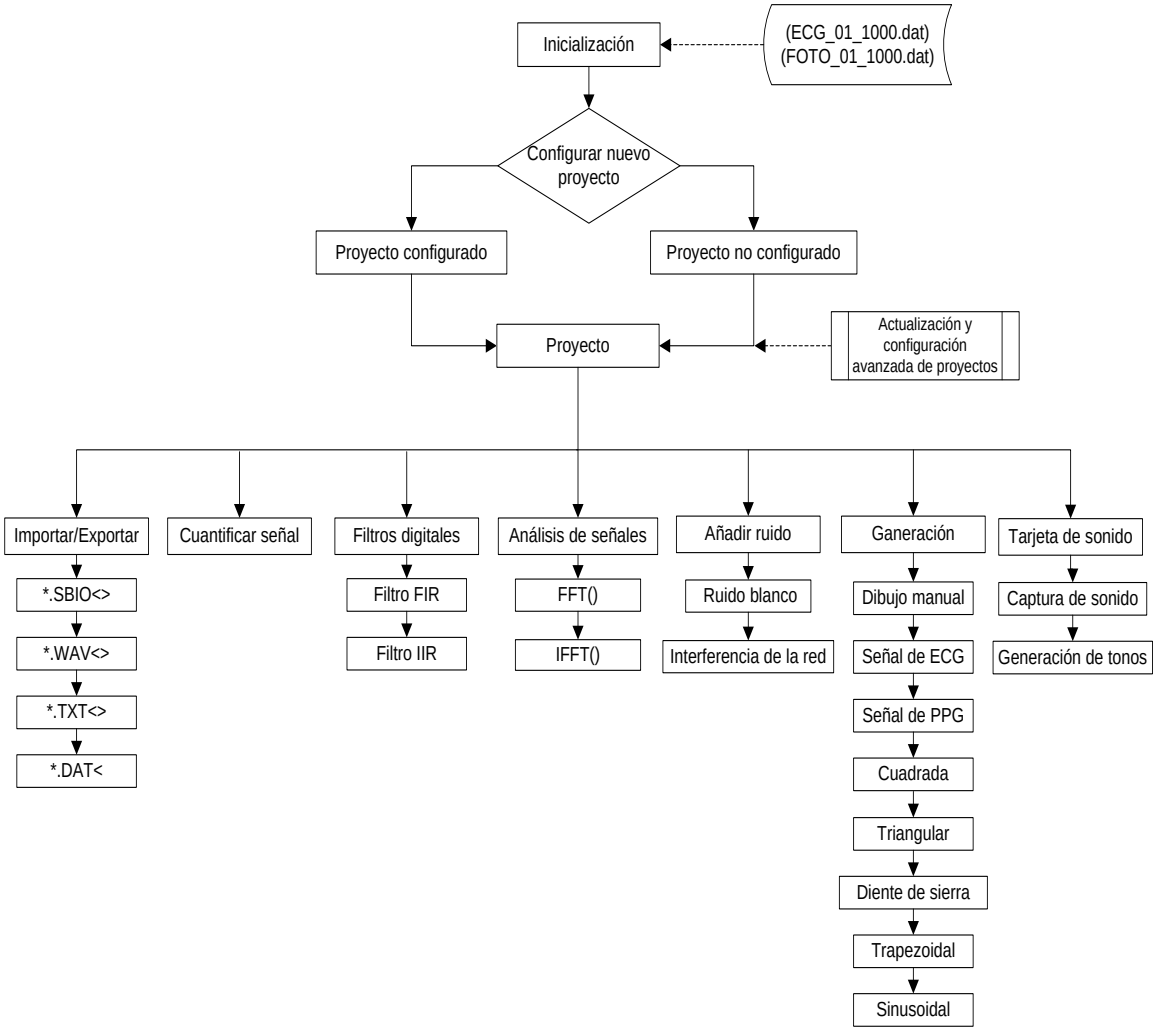
[17] Charte, Francisco. "Programación con C++Builder6®". Ed. Anaya Multimedia, 1ª edición, 1997.

[18] Gunnar Bleivik, Kjell. "C++ Builder 2010® Professional Getting started". Embarcadero Technologies™. 2010.

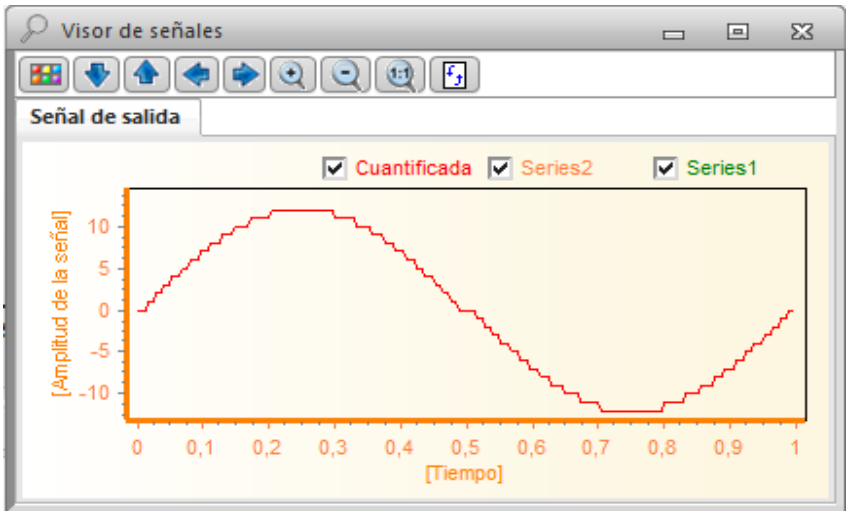
[19] Zazo Jiménez, Hector "El PC como generador de funciones usando la tarjeta PCI-DAS6025", Escola Técnica Enginyeria, Universitat Rovira i Virgili, Departament d´ Enginyeria Electrónica Eléctrica i Automática. septiembre del 2009.

ANEXOS

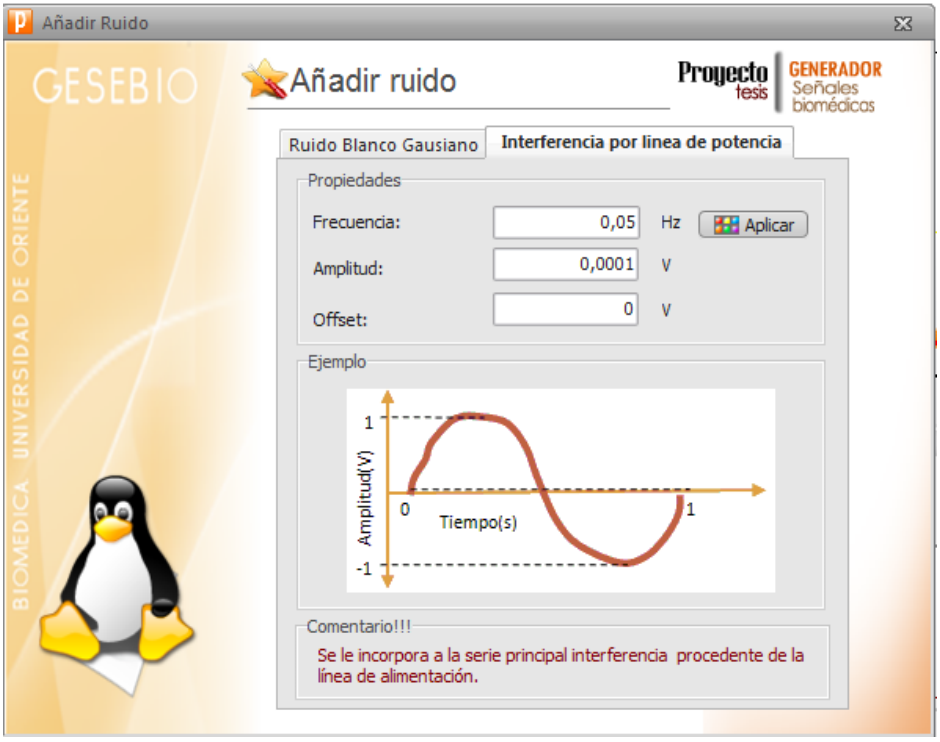
Anexo # 1. Diagrama de flujo general de la aplicación "GESEBIOv1.0".



Anexo # 2. Ventanas de la herramienta de software "GESEBIOv1.0".



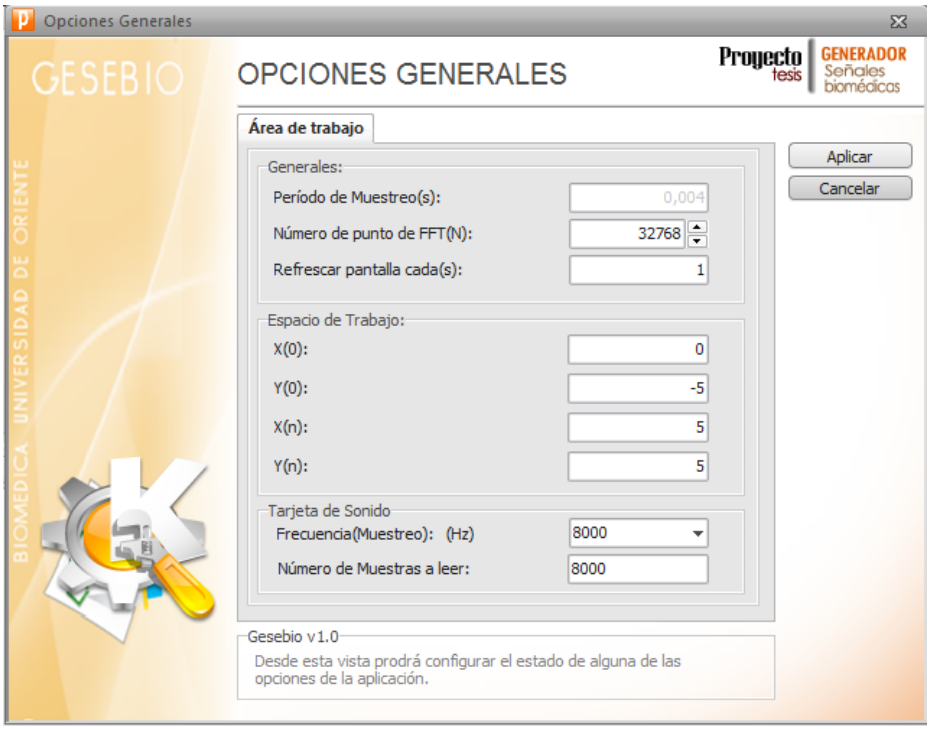
Anexo 2.1. *Visor de señales de "GESEBIOv1.0".*



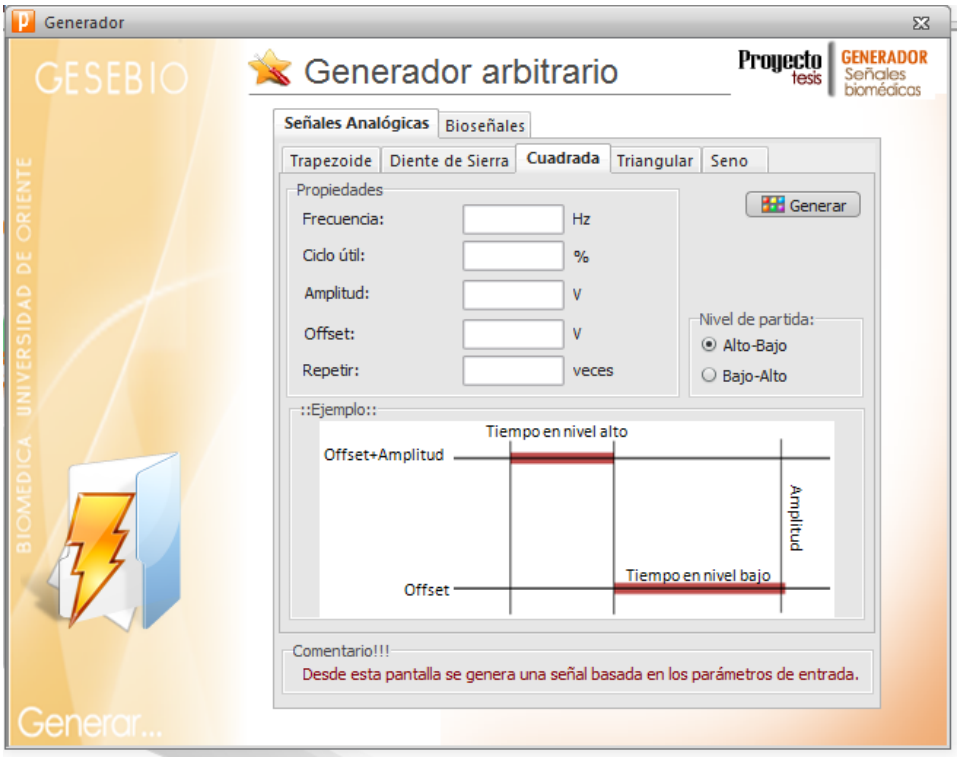
Anexo 2.2. *Generador de ruido o interferencia de "GESEBIOv1.0".*

Anexo 2.3. *Ventana de filtrado digital de "GESEBIOv1.0".*

Anexo 2.4. *Ventana de adquisición de audio de "GESEBIOv1.0".*



Anexo 2.5. *Ventana de opciones generales de "GESEBIOv1.0".*



Anexo 2.6. *Ventana de "Generación de señales" de "GESEBIOv1.0".*

Anexo #3. Expresiones empleadas para el cálculo de coeficientes de filtros IIR a partir de las aproximaciones de Butterworth.

Filtro pasa bajos:

$$\Omega_0 = tan(\frac{\omega_c}{2})$$

$$H_i(z) = \frac{G_i(1+z^{-1})^2}{1 + a_{i1}z^{-1} + a_{i2}z^{-2}}$$

$$G_i = \frac{{\Omega_0}^2}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i1} = \frac{2({\Omega_0}^2 - 1)}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i2} = \frac{1 + 2\Omega_0 cos(\Phi_i) + {\Omega_0}^2}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

Filtro pasa altos:

$$\Omega_0 = -\cot\left(\frac{\omega_c}{2}\right)$$

$$H_i(z) = \frac{G_i(1+z^{-1})^2}{1 + a_{i1}z^{-1} + a_{i2}z^{-2}}$$

$$G_i = \frac{{\Omega_0}^2}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i1} = \frac{2({\Omega_0}^2 - 1)}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i2} = \frac{1 + 2\Omega_0 cos(\Phi_i) + {\Omega_0}^2}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

Filtro pasa banda:

$$c = \frac{sin(\omega_{pa} + \omega_{pb})}{sin(\omega_{pa}) + sin(\omega_{pb})}$$

$$\Omega_0 = abs(\frac{c - cos(\omega_{pb})}{sin(\omega_{pb})})$$

$$H_i(z) = \frac{G_i(1+z^{-2})^2}{1 + a_{i1}z^{-1} + a_{i2}z^{-2} + a_{i3}z^{-3} + a_{i4}z^{-4}}$$

$$G_i = \frac{{\Omega_0}^2}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i1} = \frac{4c(\Omega_0 cos(\Phi_i) - 1)}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i2} = \frac{2(2c^2 + 1 - {\Omega_0}^2)}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i3} = -\frac{4c(\Omega_0 cos(\Phi_i) - 1)}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i4} = \frac{1 + 2\Omega_0 cos(\Phi_i) + {\Omega_0}^2}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

Filtro rechaza banda:

$$c = \frac{sin(\omega_{pa} + \omega_{pb})}{sin(\omega_{pa}) + sin(\omega_{pb})}$$

$$\Omega_0 = abs(\frac{sin(\omega_{pb})}{cos(\omega_{pb}) - c})$$

$$H_i(z) = \frac{G_i(1 - 2cz^{-1} + z^{-2})^2}{1 + a_{i1}z^{-1} + a_{i2}z^{-2} + a_{i3}z^{-3} + a_{i4}z^{-4}}$$

$$G_i = \frac{{\Omega_0}^2}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i1} = \frac{4c(\Omega_0 cos(\Phi_i) - \Omega_0)}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i2} = \frac{2(2c^2{\Omega_0}^2 + {\Omega_0}^2 - 1)}{1 - 2\Omega_0 cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i3} = -\frac{4c\Omega_0(cos(\Phi_i) + \Omega_0)}{1 - 2\Omega_0\cos(\Phi_i) + {\Omega_0}^2}$$

$$a_{i4} = \frac{1 + 2\Omega_0 cos(\Phi_i) + {\Omega_0}^2}{1 - 2\Omega_0 cos(\Phi_i) + {\Omega_0}^2}$$

Filtro Notch

$$b = \frac{1}{1 - tan(\frac{\Delta\omega}{2})}$$

$$H(z) = \frac{1 - 2\,cos(\omega_0)z^{-1} + z^{-2}}{1 - 2b\,cos(\omega_0)z^{-1} + (2b-1)z^{-2}}$$

$Para\ i = 1, 2, \dots, N.$
$Donde (N) es\ el\ orden\ del\ filtro.$

Anexo #4. Ecuaciones empleadas para el cálculo de los coeficientes de filtros FIR ideales y expresiones de ventanas.

Filtro pasa bajos:

$$h_d(n) = \frac{\sin(n\omega_c)}{n\pi} \qquad para\ n \neq 0$$

$$h_d(0) = \frac{\omega_c}{\pi} \qquad para\ n = 0$$

Filtro pasa altos:

$$h_d(n) = -\frac{\sin(n\omega_c)}{n\pi} \qquad para\ n \neq 0$$

$$h_d(0) = 1 - \frac{\omega_c}{\pi} \qquad para\ n = 0$$

Filtro pasa banda:

$$h_d(n) = \frac{\sin(n\omega_{c2}) - \sin(n\omega_{c1})}{n\pi} \qquad para\ n \neq 0$$

$$h_d(0) = \frac{\omega_{c2} - \omega_{c1}}{\pi} \qquad para\ n = 0$$

Filtro rechaza banda:

$$h_d(n) = -\frac{\sin(n\omega_{c2}) - \sin(n\omega_{c1})}{n\pi} \qquad para\ n \neq 0$$

$$h_d(0) = 1 - \frac{\omega_{c2} - \omega_{c1}}{\pi} \qquad para\ n = 0$$

Tipo de ventana	***Espresión***
Ventana rectangular	$w(n) = 1$
Ventana Hanning	$w(n) = 0.5 + 0.5\cos(\frac{2\pi n}{N})$
Ventana Hamming	$w(n) = 0.5 + 0.5\cos(\frac{2\pi n}{N})$
Ventana Blackman	$w(n) = 0.42 + 0.5\cos\left(\frac{2\pi n}{N}\right) + 0.08\cos(\frac{4\pi n}{N})$

Expresiones de Ventanas.