ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

Implementación de la transformada rápida de Fourier con NIOS II y presentación de la misma en monitor VGA

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

José Israel Mayorga Bayas

Cristóbal Andrés Candel Layana

GUAYAQUIL – ECUADOR

2012 - 2013

AGRADECIMIENTOS

Agradezco a mis padres José y Anita, por todo el apoyo brndado a lo largo de mi vida, a Evelyn y Shirley por ser un buen ejemplo para mi, a mis hermanitos Mati, José y a Estefano por ser los mejores niños del mundo, a Daniel, Manuel, Lalo, Cleo y a Ludo por mostrarme lo que es ser responsable, a mis compañeros de la universidad por el apoyo mutuo durante toda la experiencia universitaria, a mi compañero Cristóbal Candel por las horas de dedicación brindadas para que este proyecto salga adelante, finalmente a todos los profesores que tuve, gracias por compartir conocimiento y experiencia, a todos ellos gracias.

José Israel Mayorga Bayas

Es importante reconocer la ayuda prestada por los profesores que nos guiaron en el proyecto, el Ing. Ronald Ponguillo y la Ing. Maria Antonieta Álvarez, quienes estuvieron siempre dispuestos a resolver cualquier duda y compartir la información necesaria. También hago extensivo este agradecimiento a mi compañero de proyecto, Israel Mayorga por el apoyo y la dedicación brindada. Y sobre todo agradezco a Dios que nos brinda la sabiduría para conducirnos en la forma correcta.

Cristóbal Andrés Candel Layana

DEDICATORIAS

Dedicado a José y Anita por todos estos años de incondicional apoyo.

José Israel Mayorga Bayas

Todo el esfuerzo y empeño puesto en este trabajo es dedicado a mi fmilia y a todas las personas que siempre me apoyaron y me alentaron en el transcurso de mi carrera universitaria y de mi vida en general. Pero en especial a mi madre, quien día a día se preocupó por mi bienestar y se esforzó por darme todo lo que necesito.

Cristóbal Andrés Candel Layana

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Ponguillo

PROFESOR DEL SEMINARIO DE GRADUACIÓN

MSc. María Antonieta Álvarez Villanueva

PROFESOR DELEGADO POR EL DECANO DE LA FACULTAD

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de este trabajo, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”. (Reglamento de exámenes y títulos profesionales de la ESPOL)

José Israel Mayorga Bayas

Cristóbal Andrés Candel Layana

RESUMEN

El diseño e implementación de circuitos digitales constituye una parte importante en el desarrollo de la tecnología y avances en los estudios de ingeniería. En la actualidad, la evolución de estos dispositivos ha llevado a que la inclusión de los mismos en diferentes artefactos de uso común sea notoria. Debido a la capacidad de procesamiento, la versatilidad que presentan, su excelente rendimiento, su bajo costo, corto tiempo de fabricación y la confiabilidad que brindan, las FPGAs se están convirtiendo en una alternativa para crear sistemas robustos utilizando menos elementos en el diseño e implementación de los circuitos digitales y con la capacidad de cambiar y mejorar el circuito sin la necesidad de mayores cambios en el diseño electrónico. Se han vuelto dispositivos de mucha importancia en los programas de estudio de ingeniería y son sin duda parte fundamental y necesaria en la formación básica del estudiante de ingeniería.

Actualmente las FPGAs representan dispositivos de fácil manejo, con lenguajes de programación amigables, precio accesible y de fácil adquisición. Una vez que las FPGAs estuvieron al alcance de las universidades, muchas

de ellas empezaron a adecuar sus pensums académicos para incorporar como parte de sus cursos el diseño de circuitos basados en FPGA.

La transformada rápida de Fourier cobra un papel importante en las telecomunicaciones, ya que nos muestra el comportamiento en frecuencia de las señales, para posteriormente poder realizar un análisis o procesamiento de una manera rápida. Las aplicaciones de la transformada de Fourier son muchas y muy variadas como por ejemplo en el diseño de dispositivos de electrónica digital para el reconocimiento de voz e imagen, análisis espectral de todo tipo de señales, resolución de sistemas físicos y matemáticos mediante el procesamiento veloz de ecuaciones diferenciales, diseño de filtros digitales aplicados a los sistemas de telecomunicaciones, entre otros.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

ABEL	Advanced Boolean Expression Language
ADC	Analog-to-Digital Converter
ASIC	Application Specific Integrated Circuit
CMOS	Complementary metal–oxide–semiconductor
DE2	Development and Education Board
ECJ	Eclipse *Compiler* for Java
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
GUIDE	Graphical User Interface Development Environment
HDL	Hardware Description Language
IEEE	Institute of Electrical and Electronics Engineers
JTAG	Joint Test Action Group

LE	Logic Element
PCM	Pulse Code Modulation
RS-232	Recommended Standard 232
SBT	Software Build Tool
SDRAM	Synchronous Dynamic Random – Access Memory
SOPC	System on a Programmable Chip
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver-Transmitter
VGA	Video Graphics Adapter
VHDL	Very High Speed Hardware Description Languaje
VHSIC	Very High – Speed Integrated Circuit

INTRODUCCIÓN

El proyecto consiste en la implementación de un algoritmo de la transformada rápida de Fourier (FFT – Fast Fourier Transform) utilizando la tarjeta de desarrollo ALTERA DE2, basada en un dispositivo FPGA CYCLONE II de ALTERA, junto con memorias embebidas y un PROCESADOR NIOS II SOFTCORE, que ayudó en el objetivo, el cual es calcular la transformada de Fourier de una señal muestreada y presentarla en un monitor VGA.

Para la realización del proyecto se aplican tres etapas. La primera etapa está basada en un programa de interfaz gráfica realizado en el GUIDE de MATLAB, que permite al usuario generar señales básicas (seno, coseno, etc.) y configurar sus características (amplitud y frecuencia), además de generar también las modulaciones analógicas AM (Amplitude Modulation) y FM (Frequency Modulation) y enviar a la tarjeta mediante comunicación serial la señal muestreada. La segunda etapa consiste en la recepción de estos datos muestreados en la tarjeta y el cálculo de la transformada rápida de Fourier mediante un algoritmo implementado en lenguaje C. Finalmente la tercera etapa permite mostrar la transformada en un monitor VGA.

Al proyecto se lo ha estructurado en 4 capítulos, como se explica a continuación:

En el primer capítulo, se exponen los objetivos generales y específicos del proyecto, además de especificar claramente los alcances y limitaciones.

En el segundo capítulo, se da a conocer la parte teórica, explicando los conceptos básicos de los elementos y software usados para el desarrollo del proyecto, además de conocimientos adicionales que se tuvieron en cuenta para la implementación del algoritmo de la FFT.

En el tercer capítulo, se señala como se desarrolló el proyecto, para este caso se muestra el diseño base y se señala paso a paso las distintas etapas que tuvimos que ejecutar para que pueda funcionar correctamente el algoritmo de la transformada rápida de Fourier.

En el cuarto capítulo, se muestran los resultados de distintas transformadas aplicadas a señales diferentes, se realizan comparaciones y análisis de los distintos resultados obtenidos.

Finalmente se ha hecho un análisis general del proyecto basado en los resultados, de esta manera se pueden generar las respectivas conclusiones y recomendaciones.

CAPITULO 1

1. GENERALIDADES

En este capítulo se abarca el planteamiento del proyecto, se describe cuáles son sus objetivos; así como también se analiza cuáles son los alcances y limitaciones.

1.1 Objetivos

1.1.1 Objetivos Generales

El objetivo principal del proyecto es desarrollar un código en NIOS II capaz de calcular la transformada de Fourier de una señal, mediante el algoritmo de la FFT.

1.1.2 Objetivos Específicos

- Uso y manejo de la tarjeta educativa DE2 ALTERA.

- Estudiar la eficiencia e importancia de los distintos algoritmos de la transformada rápida de Fourier.

- Utilizar la interfaz VGA de la tarjeta DE2 para la presentación de los datos obtenidos como resultado del procesamiento de la señal de entrada mediante el algoritmo de la transformada rápida de Fourier en un monitor VGA.

- Analizar la manera más eficiente de adquirir los datos de la señal de entrada al sistema. Estas alternativas pueden ser utilizar una interfaz física con un convertidor analógico – digital (ADC) o el desarrollo de una interfaz gráfica en MATLAB GUIDE que se comunica serialmente con la tarjeta DE2.

- Implementar el proyecto y obtener, a partir de los resultados, las conclusiones y recomendaciones acerca de las pruebas realizadas.

1.2 Alcance y Limitaciones del Proyecto

Entre los alcances del proyecto se tiene:

- ✓ Implementación de un algoritmo para la codificación PCM de los datos generados y muestreados que serán enviados a la tarjeta DE2.
- ✓ Comunicación serial de los datos de la señal de entrada entre el computador y la tarjeta DE2.
- ✓ Lectura de los datos de la señal de entrada por medio de un código en lenguaje C, los mismos que llegan codificados en PCM.
- ✓ Decodificación PCM de los datos recibidos en la tarjeta DE2 antes de su procesamiento.
- ✓ Procesamiento de los datos mediante el algoritmo FFT Radix–2 Cooley – Tukey.
- ✓ Los datos obtenidos del procesamiento con el algoritmo FFT son finalmente mostrados en un monitor conectado a la interfaz VGA de la tarjeta DE2 Altera.

Entre las limitaciones se tiene las siguientes:

- ✓ Debido al elevado costo de un ADC de alta velocidad hemos prescindido de su uso, ya que el objetivo de nuestro proyecto es con fines educativos y no nos limita a utilizar otros métodos alternativos de adquisición de datos.

- ✓ Al utilizar comunicación serial para adquirir los datos de la señal nos hemos encontrado con limitaciones en tiempo, ya que para obtener una mayor resolución de señal necesitamos enviar mayor cantidad de datos.
- ✓ La resolución de la señal va ligada a la frecuencia de muestreo de la misma, que en nuestro caso es proporcional a la frecuencia de la señal de entrada. Esto nos obliga a establecer rangos máximos de frecuencia de la onda que se envía.
- ✓ Debido a las características del controlador VGA de la tarjeta DE2 de Altera, la presentación de la transformada de Fourier de la señal será de baja resolución, es decir, para obtener una buena apreciación de la señal no hemos dibujado la totalidad de los puntos, ya que este controlador tiene una resolución de imagen máxima de 320x240.

CAPITULO 2

2. MARCO TEÓRICO

En este capítulo se abarcan los conceptos básicos y se hace una pequeña introducción a los programas y tecnologías que fueron de utilidad durante el desarrollo del proyecto.

2.1 Introducción a Quartus II y Lenguaje VHDL

Quartus II es una herramienta de software producida por Altera para el análisis y síntesis de diseños realizados en HDL. La interfaz gráfica del software Quartus II es amigable y sencilla, permitiendo al usuario acceder a las distintas herramientas y observar archivos relevantes como los resultados de la compilación, lista de errores y advertencias, codificación fuente, etc. Altera actualiza su software regularmente, pero la versión utilizada en nuestro proyecto es 11.1. Dentro de las funciones

que puede realizar Quartus II, encontramos el análisis de circuitos lógicos, básicamente es un simulador para circuitos lógicos. [9]

HDL es utilizado para describir, diseñar y modelar sistemas digitales. Los lenguajes de descripción de hardware básicamente permiten documentar, mediante un código, el comportamiento de un circuito electrónico, definiendo las tareas que debe realizar el sistema digital. Entre los HDLs más utilizados tenemos VHDL, Verilog y ABEL. [10]

Nuestro proyecto se implementó en VHDL, por ser el lenguaje de descripción de hardware incluido en el pensum académico, además de ser el más utilizado en el desarrollo de sistemas con FPGAs.

La palabra VHDL viene de *"VHSIC (Very High – Speed Integrated Circuit) Hardware Description Language"*. Fue originalmente diseñado por el Departamento de Defensa de los Estados Unidos de América y posteriormente transferido a la IEEE. El lenguaje se define formalmente en el estándar IEEE 1076. [10]

Figura 2-1 Pantalla de Inicio de Quartus II v11.1

En la figura 2-1 se muestra la pantalla de inicio de Quartus II v11.1, este es la versión de Quartus usada para el desarrollo de este proyecto.

2.2 INTRODUCCIÓN A ECLIPSE

Eclipse, es un software de código abierto multiplataforma. Estas plataformas son usadas generalmente para desarrollar entornos de desarrollo integrados (IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ). Cabe destacar que Eclipse comenzó como un proyecto de IBM Canadá. Fue desarrollado por OTI (Object Technology International) como reemplazo de VisualAge.

Eclipse se encuentra estructurado de la siguiente manera:

- ✓ Plataforma principal para el inicio de Eclipse y ejecución de Plug-ins.
- ✓ OSGi, que es una plataforma para bundling estándar.
- ✓ El SWT, que es un Widget Toolkit portable.
- ✓ JFace, para manejo de archivos, manejos de texto, editores de texto, etc.
- ✓ Workbench de Eclipse, para vistas, editores, perspectivas, asistentes, etc.

Este software usa lenguajes de programación como C/C++ y Phython, además de permitir trabajar con lenguajes para procesamiento de texto como LaTeX, aplicaciones de red como Telnet y Sistema de gestión de base de datos, además de proveer al programador con frameworks para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc.

La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular".

2.3 NIOS II SBT PARA ECLIPSE

Nios II SBT para Eclipse es una pequeña capa de interfaz gráfica de usuario que ayuda a presentar un entorno unificado de desarrollo, El SBT para Eclipse proporciona una plataforma de desarrollo coherente que funciona para todos los procesadores Nios II.

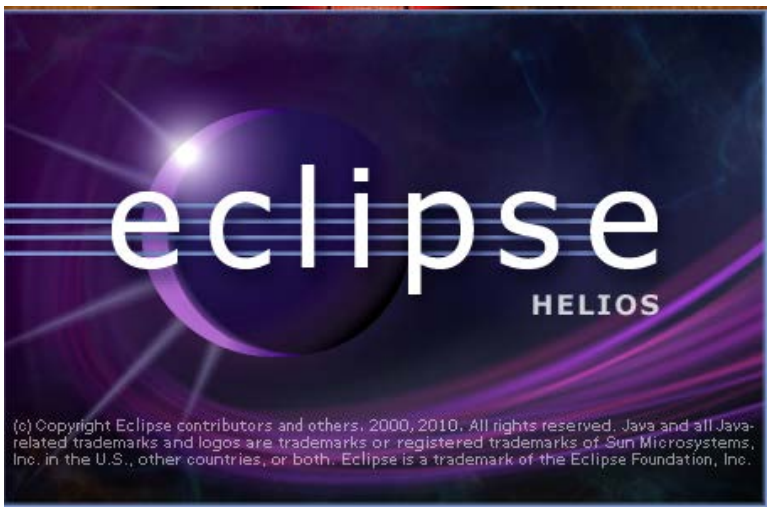


Figura 2-2 Nios II 11.1 Software Build Tools for Eclipse

En la figura 2-2 se muestra la pantalla de inicio de Nios II 11.1 Software Build Tools for Eclipse, con esta herramienta se puede llevar a cabo todas las tareas de desarrollo de software dentro de Eclipse, incluyendo creación, edición, construcción, funcionamiento, depuración y perfilado de programas. Nios II SBT para Eclipse se basa en la estructura de Eclipse Framework y Eclipse Development Toolkit (CDT) Plugins.

2.4 TARJETA DE DESARROLLO ALTERA DE2

La tarjeta de desarrollo de Altera DE2 mostrada en la figura 2-3, fue diseña con fines educativos, creada por profesores, dirigida para profesores, investigadores y estudiantes. [11]

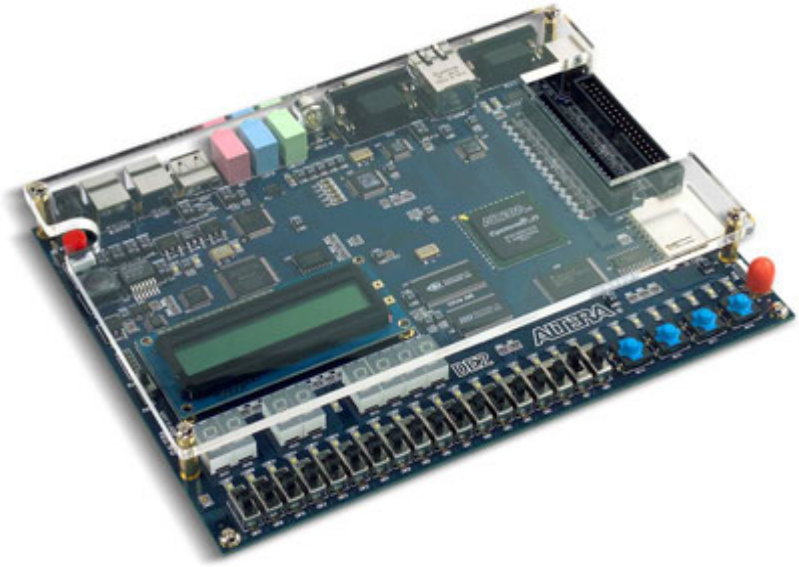


Figura 2-3 Tarjeta DE2 de ALTERA [1]

Ideal para la fácil comprensión y aprendizaje interactivo de la lógica digital, organización computarizada y FPGAs. Adecuada para realizar un amplio número de ejercicios, desde tareas simples para ilustrar conceptos fundamentales de cursos de lógica digital, hasta la implementación de diseños avanzados. Todas estas características convierten a la tarjeta DE2 de Altera en una herramienta indispensable en los laboratorios de las universidades del mundo. [11]

El kit de la Tarjeta de Desarrollo Altera DE2 posee [12]:

- ✓ Una tarjeta DE2 de 8'' x 6'' con un FPGA Cyclone II EP2C35.
- ✓ Adaptador de 9V AC/DC
- ✓ Cable USB para programar y controlar la FPGA
- ✓ Cobertor para la tarjeta
- ✓ Guía de Instalación
- ✓ CD con documentación de la DE2, material de apoyo e instaladores de los programas Quartus II y NIOS II.

La tarjeta DE2 está fabricada siguiendo los mismos estándares de diseño estricto que se utilizan en la construcción masiva de productos de gama alta como las potentes motherboards de los computadores y demás equipos electrónicos de calidad. Los distintos componentes que conforman la tarjeta DE2 fueron seleccionados para brindar el mejor desempeño al usuario. Se consideraron protecciones en la fuente de poder para evitar los accidentes más comunes causados por problemas en la alimentación de energía eléctrica. [13]

Esta tarjeta ofrece a los usuarios muchas características que permiten el desarrollo de distintos proyectos. La selección de cada uno de los distintos componentes se hizo de acuerdo a los productos con mayor captación de volumen en producción multimedia.

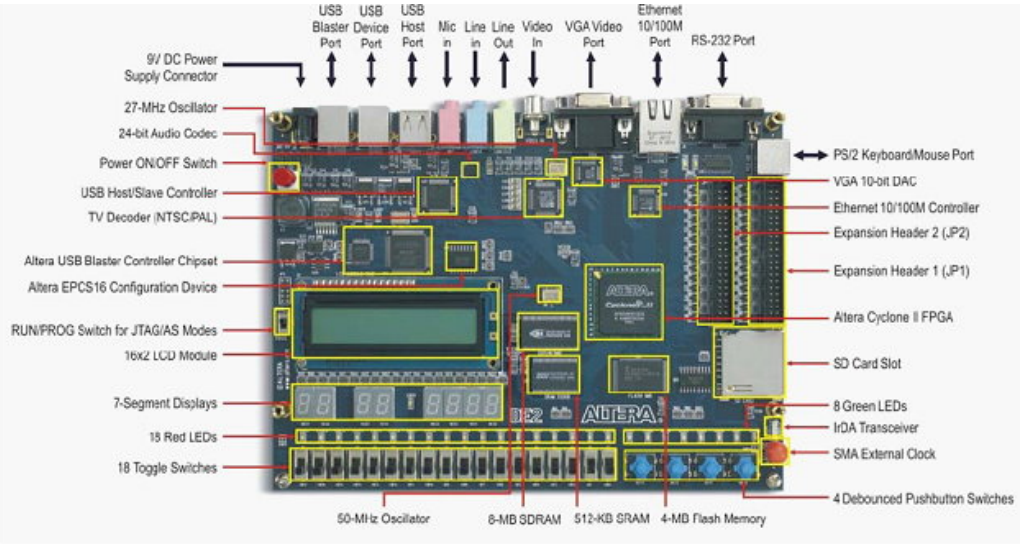


Figura 2-4 Periféricos y E/S, Tarjeta DE2 Altera [2]

Como se observa en la figura 2-4 los periféricos incluidos en la tarjeta DE2 son [2]:

- ✓ FPGA Altera Cyclone II 2C35 con 35000 Les
- ✓ Altera Serial Configuration device (EPCS16) para Cyclone II 2C35
- ✓ USB Blaster para programar y usar la tarjeta
- ✓ JTAG
- ✓ 8 Mbyte (1M x 4 x 16) SDRAM
- ✓ 1 Mbyte Flash Memory
- ✓ Sócalo para tarjeta SD
- ✓ 4 Push – button

- ✓ 18 switches DPDT
- ✓ 9 LEDs verdes
- ✓ 18 LEDs rojos
- ✓ Oscilador de 50 MHz para señal de reloj
- ✓ Audio Codec y Jack para micrófono
- ✓ Conector VGA
- ✓ Decoder y conector de TV
- ✓ 10/100 Ethernet controller con sócalo
- ✓ Controlador USB
- ✓ Transceiver RS-232
- ✓ Conector para mouse y teclado
- ✓ Transciever IrDA
- ✓ Dos puertos de expansión de 40 pines

2.5 FPGA

Una FPGA es un dispositivo semiconductor que contiene bloques lógicos, cuya interconexión y funcionalidad se puede programar. Básicamente, la FPGA es un dispositivo lógico que contiene un arreglo de celdas lógicas y switches programables. Una celda lógica puede ser configurada o programada para que realice una función, mientras que la

disposición de los switches programables se puede distribuir de manera que proporcione conexión entre celdas lógicas.

La implementación de un diseño en FPGA consiste en especificar la función o tarea de cada celda lógica y seleccionar la disposición más adecuada de los switches programables. Luego este diseño se carga sobre la FPGA y se obtiene un circuito personalizado. [14]

Figura 2-5 Cyclone II FPGA [3]

En la figura 2-5 muestra una FPGA de la familia Cyclone II, esta es la usada en la tarjeta De2 de Altera.

Una de las mayores ventajas de las FPGAs es justamente su capacidad de ser programables permitiendo al usuario construir circuitos a su gusto. A diferencia de los ASICs, las FPGAs son más versátiles y flexibles, ya que el proceso de creación del sistema digital se realiza en

el campo (de ahí su nombre “field programmable”) en lugar de diseñarse en una fábrica.

Los principales beneficios de los FPGAs son:

- ✓ Rendimiento
- ✓ Bajo precio
- ✓ Confiabilidad
- ✓ Mantenimiento a largo plazo

Hoy las FPGAs están presentes en campos tan diversos como la electrónica de consumo, la investigación científica, entretenimiento, etc. Tiene cabida en empresas dedicadas a la fabricación de sistemas de seguridad, climatización, telecomunicaciones, control automático industrial, etc.

2.6 PROCESADOR EMBEBIDO NIOS II

El procesador NIOS II de Altera es un procesador soft – core, que a diferencia de los procesadores fijos prefabricados, es descrito mediante un código en HDL y luego cargado sobre una FPGA. De esta manera el procesador NIOS II ofrece mayor flexibilidad, ya que puede ser configurado y mejorado añadiendo o removiendo características en el sistema. [15]

El Procesador Embebido Nios II es un sistema flexible capaz de ajustarse a las necesidades del diseñador, esta flexibilidad es gracias a la tecnología del SOPC builder que tiene como ventaja integrar en un mismo chip una o varios CPUs y sus periféricos asociados, lo cual optimiza el sistema.

SOPC builder brinda un entorno específico, permitiendo la definición y configuración a medida del microprocesador NIOS II y que por medio de la herramienta Quartus II puede ser directamente utilizado sobre el FPGA.

NIOS II es un procesador muy eficiente debido a las bondades que proporciona, su característica principal es que posee un grupo de registros de propósito general de 32 bits.

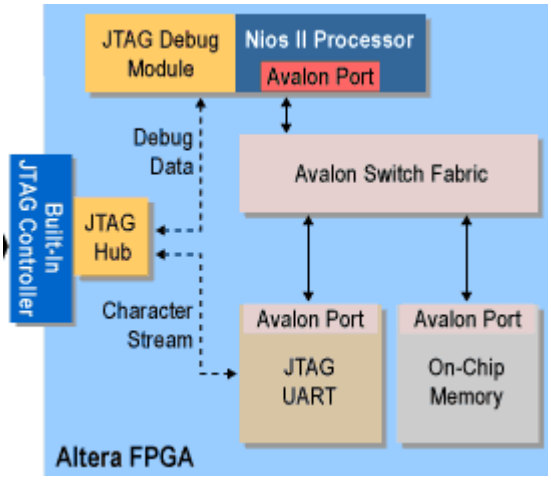


Figura 2-6 Sistema Elemental con NIOS II System [4]

La figura 2-6 muestra un sistema elemental con NIOS II, que está compuesto por: el núcleo procesador NIOS II, memoria interna de programa y de datos, periféricos integrados e interfaces para memoria externa y/o entrada/salida.

Existen tres versiones del procesador Nios II, las cuales tienen la misma arquitectura de instrucciones de 32 bits:

- ✓ NIOS II /f (rápido): Es de alto rendimiento, con *pipeline* de 6 etapas aumenta su desempeño con opciones específicas como: memorias caché de instrucciones y datos o una unidad de manejo de memoria.

- ✓ NIOS II /s (estándar): Tiene un *pipeline* de 5 etapas que con una unidad aritmética lógica busca combinar rendimiento y consumo de recursos.
- ✓ NIOS II /e (económico): Requiere de menos recursos de la FPGA, no posee *pineline* y es muy limitada porque carece de las operaciones de multiplicación y división.

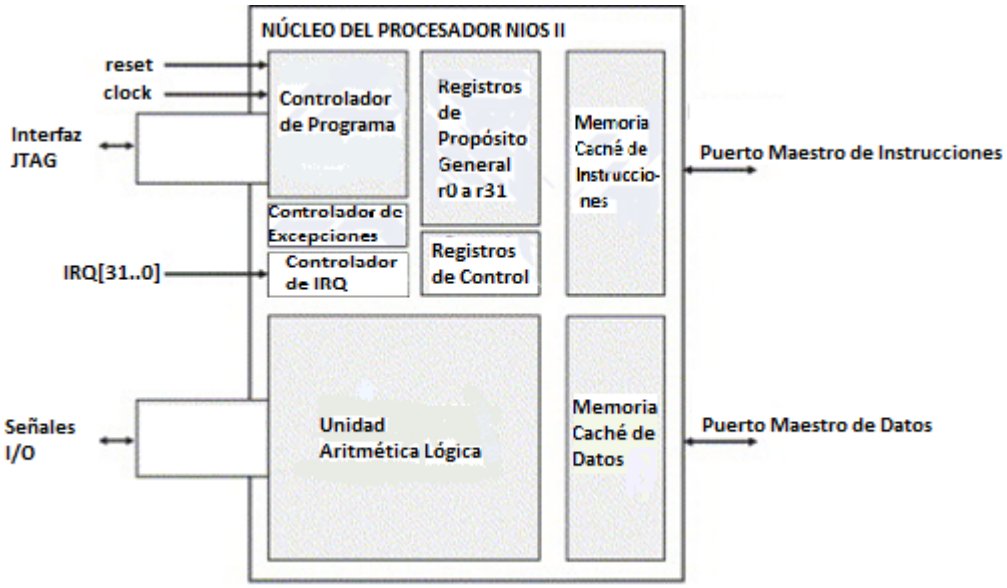


Figura 2-7 Diagrama de Boques del Procesador NIOS II [5]

El procesador Nios II mostrado en la figura 2-7, está formado por unidades funcionales dependiendo de la versión implementada, entre las fundamentales tenemos: registros, unidad aritmético-lógica, interfaz para instrucciones definidas por el usuario, controlador de excepciones,

bus de instrucciones, bus de datos, memoria caché de instrucciones y datos, interfaz de acoplamiento directo de memoria de instrucciones y datos y módulo de depuración JTAG.

2.7 TRANSFORMADA DE FOURIER

Figura 2-8 Matemático francés Joseph Fourier (1768-1830) [6]

La figura 2-8 muestra al Matemático francés Joseph Fourier, conocido por sus diversos trabajos en el área de las Matemáticas, de sus trabajos se conoce que la transformada de Fourier, a diferencia de la serie de Fourier que solo se puede emplear con señales periódicas, puede ser aplicada a cualquier señal que cumpla con las condiciones de Dirichlet que se describen a continuación:

- Que la señal sea absolutamente integrable, es decir:

$$\int_{-\infty}^{\infty}|x(t)|^2\,dt < \infty \qquad (2.1)$$

- Que tenga un grado de oscilación finito.
- Que tenga un número finito de discontinuidades.

La transformada de Fourier es una particularización de la transformada de Laplace con $s = j\omega$, siendo $\omega = 2\pi f$, y se define como:

$$x(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}\,dt \qquad (2.2)$$

Y su inversa se define como:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} x(\omega)e^{j\omega t}\,d\omega \qquad (2.3)$$

2.8 TRANSFORMADA DISCRETA 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 *'nT_s'* siendo *'T_s'* el periodo de muestreo.

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 DFT requiere que la función de entrada sea una secuencia discreta y de duración finita. Dichas secuencias se suelen generar a partir del muestreo de una función continua, como puede ser la voz humana.

La entrada de la DFT es una secuencia finita de números reales o complejos, de modo que es ideal para procesar información almacenada en soportes digitales.

La Inversa de la Transformada Discreta de Fourier (IDFT) realiza el proceso contrario de la DFT.

Recordemos el par de ecuaciones de la DFT:

$$X(k) = \sum_{n=0}^{N-1} x(n) W^{nk} \; ; \qquad k = 0,1, \dots, N-1 \qquad \textit{(2.4)}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W^{-nk} \; ; \qquad n = 0,1, \dots, N-1 \qquad \textit{(2.5)}$$

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

$$W = e^{-j2\pi/N} \qquad \textit{(2.6)}$$

Observar que *'W'* es una función de longitud *N*, por ello, también suele expresarse como W_N.

El inconveniente de realizar unos algoritmos que implementen tal cual estas fórmulas es la cantidad de tiempo requerido para computar la salida. Esto es debido a que los índices *k* y *n* deben variar de *0* a *N-1* para conseguir el rango de salida completo y, por tanto, se deben realizar N^2 operaciones.

En particular, la DFT se utiliza comúnmente en procesado digital de señales y otros campos relacionados dedicados a analizar las frecuencias que contiene una señal muestreada, también para resolver ecuaciones diferenciales parciales, y para llevar a cabo operaciones como convoluciones o multiplicaciones de enteros largos.

Un factor muy importante para este tipo de aplicaciones es que la DFT puede ser calculada de forma eficiente en la práctica utilizando el algoritmo de la transformada rápida de Fourier o FFT (Fast Fourier Transform).

Los algoritmos FFT se utilizan tan habitualmente para calcular DFTs que el término "FFT" muchas veces se utiliza en lugar de "DFT" en lenguaje coloquial. Formalmente, hay una diferencia clara: "DFT" hace alusión a una transformación o función matemática, independientemente de cómo se calcule, mientras que "FFT" se refiere a una familia específica de algoritmos para calcular DFTs.

2.9 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:

$$X(k) = x(0) + x(1)W^k + x(2)W^{2k} + \cdots + x(N-1)W^{(N-1)k} \quad (2.7)$$

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.

La FFT aprovecha la periodicidad y simetría del factor twiddle *'W'* para el cálculo del Transformada Discreta de Fourier. La periodicidad de *'W'* implica:

$$W^k = W^{k+N} \qquad (2.7)$$

y su simetría implica:

$$W^k = -W^{k+N/2} \qquad (2.8)$$

La FFT descompone la DFT de *N* puntos en transformadas más pequeñas. Una DFT de *N* puntos es descompuesta en dos DFT's de *(N/2)* puntos. Cada DFT de *(N/2)* puntos se descompone a su vez en dos DFT's de *(N/4)* puntos y así sucesivamente. Al final de la descomposición se obtienen *(N/2)* DFT´s de *2* puntos cada una. La transformada más pequeña viene determinada por la base de la FFT.

2.10 ALGORITMO COOLEY-TUKEY (RADIX-2)

El algoritmo propuesto por Cooley y Tukey en 1965 permite realizar el cálculo de la DFT de una forma más eficiente haciendo uso del enfoque divide y vencerás. En las siguientes subsecciones se presentarán dos versiones del algoritmo radix-2, una implementación específica del algoritmo, orientada al cálculo eficiente de secuencias de datos de longitud 2^p, donde *P* es un número natural.

2.10.1 FFT RADIX-2 DIF

El algoritmo radix-2 DIF (Decimation In Frequency) se obtiene al aplicar el enfoque divide y vencerás en la definición de la DFT. De esta forma es posible dividir la sumatoria en dos sumatorias de longitud *N/2* de la siguiente forma:

$$X(k) = \sum_{n=0}^{(N/2)-1} x(n) {W_N}^{kn} + \sum_{n=N/2}^{N-1} x(n) {W_N}^{kn} \quad (2.9)$$

donde

$$W_N = e^{j2/N} \qquad (2.10)$$

y W_N es llamado factor de fase o factor Twiddle como se mencionó anteriormente.

Si se cambia el índice de la segunda sumatoria en la ecuación 2.9 y considerando que $W_N^{kN/2} = (-1)^k$, se obtiene:

$$X(k) = \sum_{n=0}^{(N/2)-1} \left[x(n) + (-1)^k \, x(n + \tfrac{N}{2}) \right] {W_N}^{kn} \quad (2.11)$$

asumiendo valores pares de *k* y recordando que

$$(W_N)^2 = W_{N/2}$$

$$X(2k) = \sum_{n=0}^{(N/2)-1} f_{par} \, (W_{N/2})^{nk} ; \quad k = 0,1, \dots, \frac{N}{2} - 1 \quad (2.12)$$

por otra parte, para valores de *k* impares se tiene que:

$$X(2k+1) = \sum_{n=0}^{(N/2)-1} f_{impar} \, (W_{N/2})^{nk} ; \quad k = 0,1, \dots, \frac{N}{2} - 1 \quad (2.13)$$

donde se obtiene que:

$$f_{par} = x(n) + x(n + \tfrac{N}{2}) \qquad (2.14)$$

$$f_{impar} = \left[x(n) + x(n + \frac{N}{2})\right] W_N{}^n \quad (2.15)$$

El algoritmo propuesto puede utilizarse de forma recursiva reduciendo a la mitad cada vez el número de puntos de la transformada hasta reducirse a una DFT de 2 puntos. El hecho que se obtengan los valores de la transformada para frecuencias pares e impares de forma separada a partir de la secuencia de datos ordenada conlleva al uso de un algoritmo de ordenamiento de los datos resultantes, es esta característica la que le da su nombre al algoritmo.

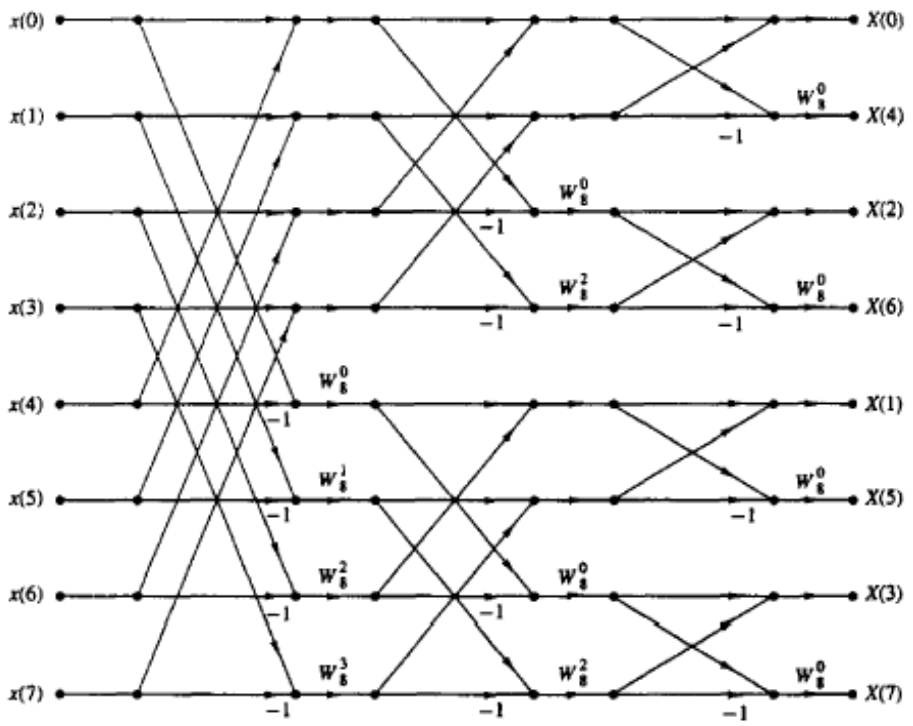


Figura 2-9 Representación gráfica del algoritmo DIF radix-2 [7]

En la figura 2.9 se muestra una representación gráfica del algoritmo para *N = 8*, puede observarse como a partir de una secuencia de datos ordenada se realizan una serie de operaciones mariposa para finalmente obtener los valores de DFT en orden de bit invertido.

2.10.2 FFT RADIX-2 DIT

El algoritmo radix-2 DIT (Decimation In Time) es similar conceptualmente al DIF discutido anteriormente, con la diferencia que en este caso se aplica el concepto divide y vencerás a la secuencia de datos de entrada. De esta forma dada una secuencia de datos *x(n)* con $x_p(n) = x(2n')$ y $x_i(n) = x(2n' + 1)$ las correspondientes secuencias de elementos pares e impares de *x(n)*, respectivamente, donde $n' = 0, 1, \ldots, N/2 - 1$ se tiene que:

$$X(k) = \sum_{n=0}^{(N/2)-1} x_p(n) {W_{N/2}}^{kn} + {W_N}^k \sum_{n=0}^{(N/2)-1} x_i(n) {W_{N/2}}^{kn} \quad (2.16)$$

En la ecuación 2.16 se puede observar que cada sumatoria corresponde a la transformada discreta de la función par/impar, por lo cual puede reescribirse como:

$$X(k) = X_p(k) + {W_N}^k X_i(k)\,; \qquad k = 0,1,\ldots,\frac{N}{2} - 1 \qquad (2.17)$$

Como es posible observar en la ecuación 2.17, el cálculo de la FFT DIT tal y como se ha expresado solamente suministra los valores de DFT

para *N/2* muestras de datos, sin embargo, considerando que $X_p(k)$ y $X_i(k)$ tienen periodo *N/2* y utilizando la propiedad del factor de fase $W_N^{k+N/2} = -W_N^k$ se tiene que:

$$X(k + N/2\,) = X_p(k) - W_N{}^k X_i(k)\,; \quad k = 0,1,2,\ldots,\frac{N}{2} - 1 \quad \textit{(2.18)}$$

De esta forma las ecuaciones 2.17 y 2.18 suministran la transformada discreta de todos los *N* valores de entrada.

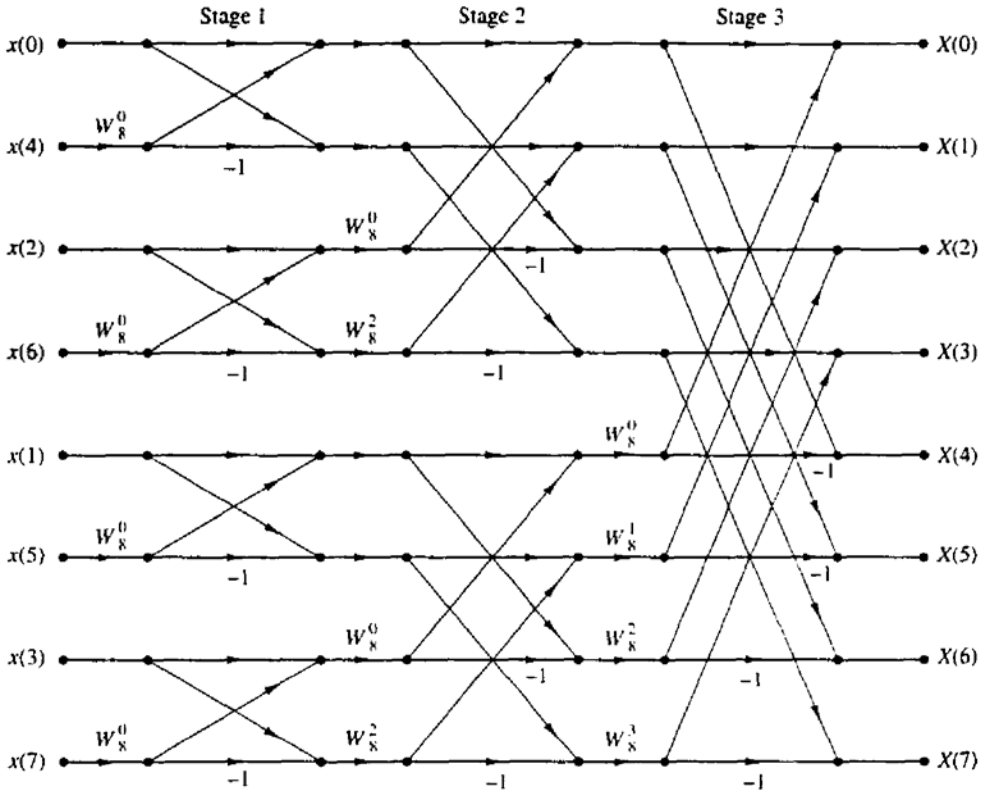


Figura 2-10 Representación gráfica del algoritmo DIT radix-2 [8]

En la figura 2.10 se muestra un gráfico del algoritmo aplicado a una secuencia de *N = 8* muestras, como es posible observar, los valores de entrada son reordenados mientras que la salida en frecuencia se obtiene automáticamente en orden, es por esta razón que se le da el nombre DIT al algoritmo.

2.11 INTRODUCCION A MATLAB

MATLAB® es un lenguaje de alto nivel y un entorno interactivo para el cálculo numérico, la visualización y la programación. Mediante MATLAB, es posible analizar datos, desarrollar algoritmos y crear modelos o aplicaciones. Las herramientas y las funciones matemáticas incorporadas permiten explorar diversos enfoques y llegar a una solución antes que con hojas de cálculo o lenguajes de programación tradicionales, como pueden ser C/C++ o Java™.

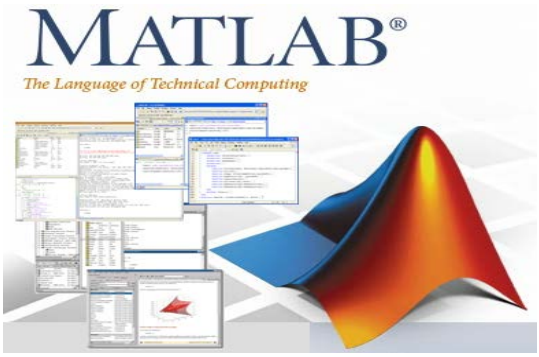


Figura 2-11 Pantalla de Inicio de Matlab

La figura 2-11 muestra la pantalla de inicio de Matlab, esta herramienta se puede utilizar en una gran variedad de aplicaciones, tales como procesamiento de señales y comunicaciones, procesamiento de imagen y vídeo, sistemas de control, pruebas y medidas, finanzas computacionales y biología computacional.

Más de un millón de ingenieros y científicos de la industria y la educación utilizan MATLAB, el lenguaje del cálculo técnico. [15]

Para la realización de este proyecto se ha hecho uso de la herramienta GUIDE de MATLAB para crear una interfaz gráfica en un computador.

CAPITULO 3

3. DISEÑO E IMPLEMENTACION

En este capítulo se explican los detalles del diseño del proyecto mediante la elaboración de un diagrama de bloques que muestra el funcionamiento básico del proyecto.

3.1 Diagrama de bloque

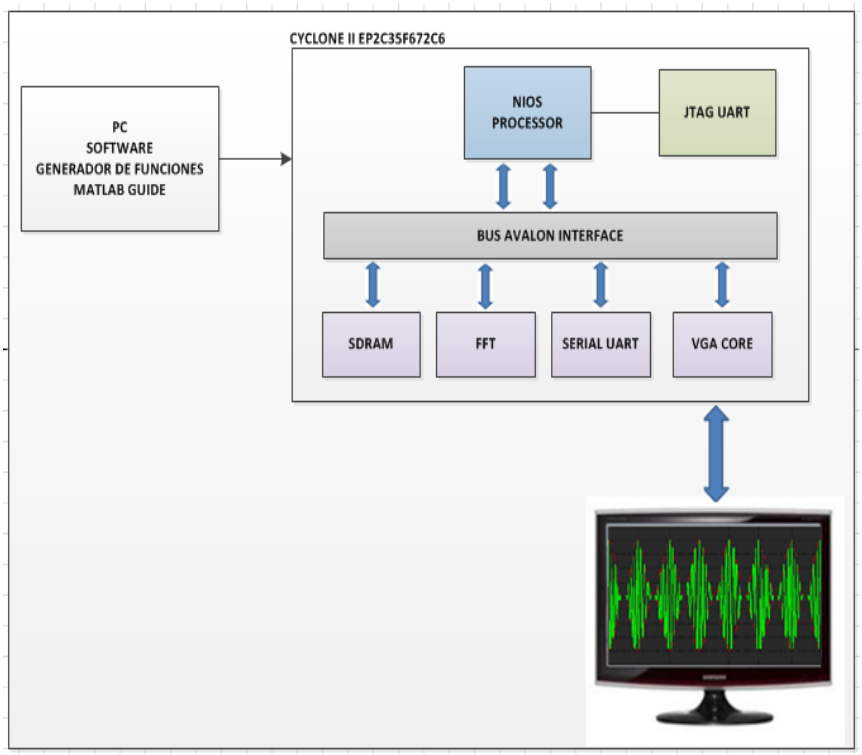


Figura 3-1 Diagrama de bloques del sistema

La figura 3.1 muestra el diagrama de bloques del sistema, el cual consta de tres etapas fundamentales:

- **Software Generador de Funciones.-** Esta etapa consiste de un software desarrollado en MATLAB utilizando la interfaz gráfica del mismo.
- **Máquina desarrollada en la tarjeta DE2.-** En esta etapa se describen todos los bloques de hardware utilizados en este proyecto.
- **Procesamiento y presentación de los datos en un Monitor VGA.**- En esta etapa se hace la presentación de los datos procesados con el algoritmo FFT implementado en el procesador de la tarjeta.

A continuación se explica cada etapa con mayor detalle.

3.2 Software Generador de Funciones

El software generador de funciones utilizado en el proyecto está desarrollado en la interfaz gráfica de usuario de MATLAB.

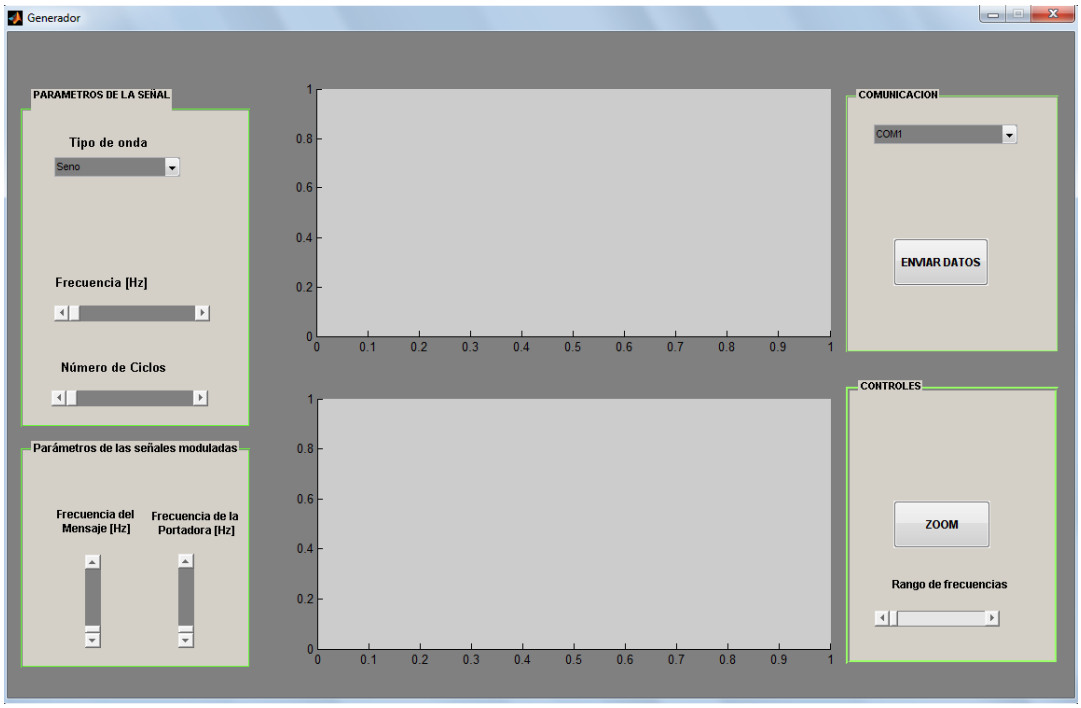


Figura 3-2 Ventana inicial del Generador de Funciones

En la figura 3.2 observamos la ventana inicial del Software Generador de Funciones, el mismo que consta de cuatro paneles de configuración de parámetros y dos ventanas de gráficos que se explican detalladamente a continuación.

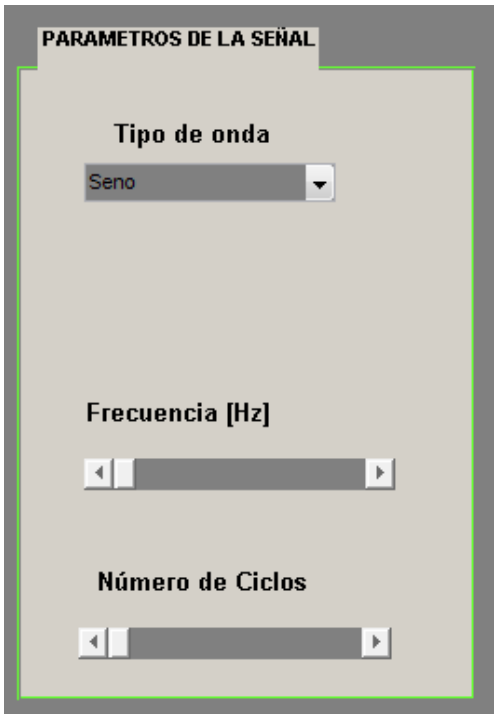


Figura 3-3 Panel "Parámetros de la Señal"

El Panel **"Parámetros de la Señal"** observado en la figura 3.3 consta de los siguientes elementos:

- **Menú Pop – up "Tipo de onda".-** Permite seleccionar el tipo de onda que se desea enviar a través de la interfaz serial hacia la tarjeta DE2 para su posterior procesamiento.

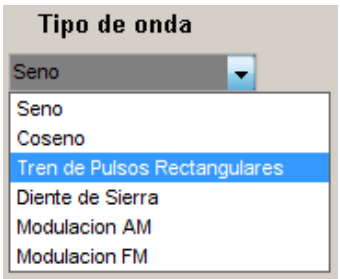


Figura 3-4 Tipos de onda

En la figura 3.4 se observan los distintos tipos de onda que podemos seleccionar en el software Generador de Funciones. Debido a que el proyecto realizado tiene fines educativos se configuraron las señales más frecuentes, seno, coseno, tren de pulsos rectangulares, diente de sierra y las modulaciones básicas, AM y FM.

- **Slider "Frecuencia".-** Por medio de este elemento de la interfaz gráfica podemos configurar la frecuencia de la señal. En la figura 3-5 se ha seleccionado una frecuencia de 240 Hz. Hay que tener en cuenta que este slider solo sirve para configurar las cuatro primeras señales que corresponden a las ondas periódicas.

Figura 3-5 Frecuencia de la señal

La frecuencia a fijar está en los rangos de 1 Hz hasta 1000 Hz, debido a que para el procesamiento óptimo de la señal con el algoritmo de la transformada rápida de Fourier se necesita que la frecuencia de muestreo sea por los menos el doble de la frecuencia de la señal, para hechos teóricos, pero en la práctica se consiguen resultados satisfactorios con frecuencias de muestreo de por los menos ocho veces la frecuencia del mensaje. Para nuestro caso se utiliza una

frecuencia de muestreo igual a 64 veces la frecuencia del mensaje. Al tener una alta frecuencia de muestreo, el periodo se reduce considerablemente, lo que implica una mayor cantidad de muestras de señal, lo que se traduce en más tiempo de demora en la transmisión serial de datos.

- **Slider "Número de Ciclos".-** Este elemento del generador de funciones permite configurar la cantidad de ciclos de señal que se dibujan en la ventana de gráficos correspondiente a la onda en el tiempo. Los rangos van de 0.1 ciclos hasta 20 ciclos de señal.

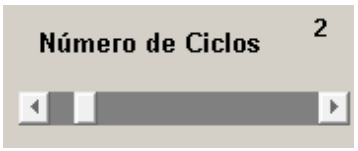


Figura 3-6 Ciclos de señal

El fin de los rangos de ciclo de señal es conseguir que el usuario tenga una mejor apreciación de la señal que se dibuja en la ventana de gráficos del generador de funciones.

Las ventanas de gráficos del generador de funciones muestran la señal en el tiempo y su correspondiente transformada de Fourier, como vemos en la figura 3-7, estos gráficos se los presenta en el software

generador de funciones con el fin de tener una idea de lo que la máquina implementada en la tarjeta DE2 debe realizar.

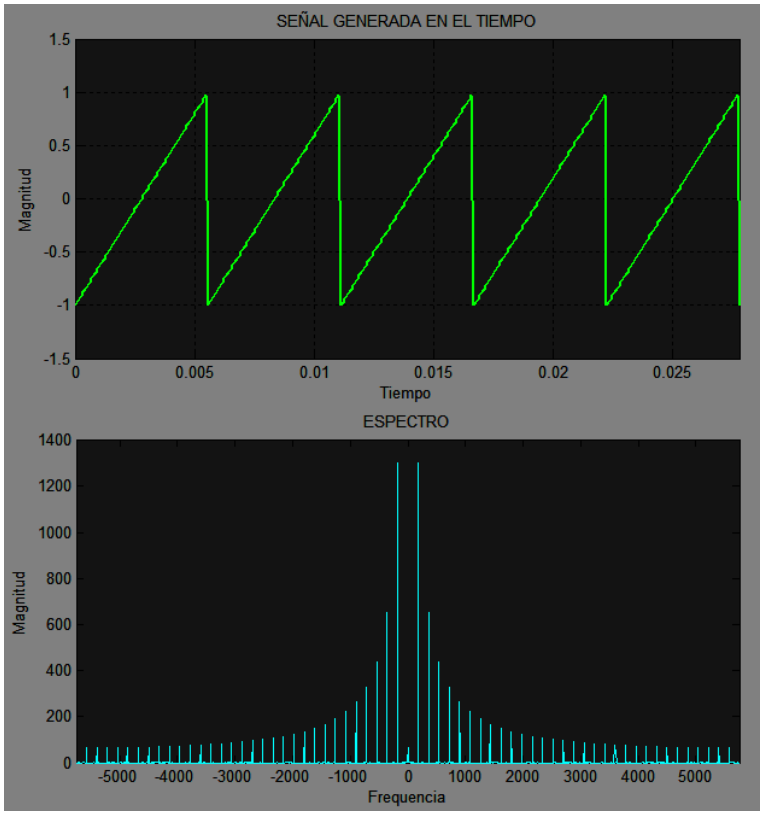


Figura 3-7 Ventanas de gráficos del generador

En la figura 3-8 observamos la señal en la ventana de gráficos con las características de tipo de onda, frecuencia y ciclos de señal configurados en los elementos descritos.

Tipo de onda: *Diente de sierra*
Frecuencia: *180 Hz*
Número de ciclos: *5*

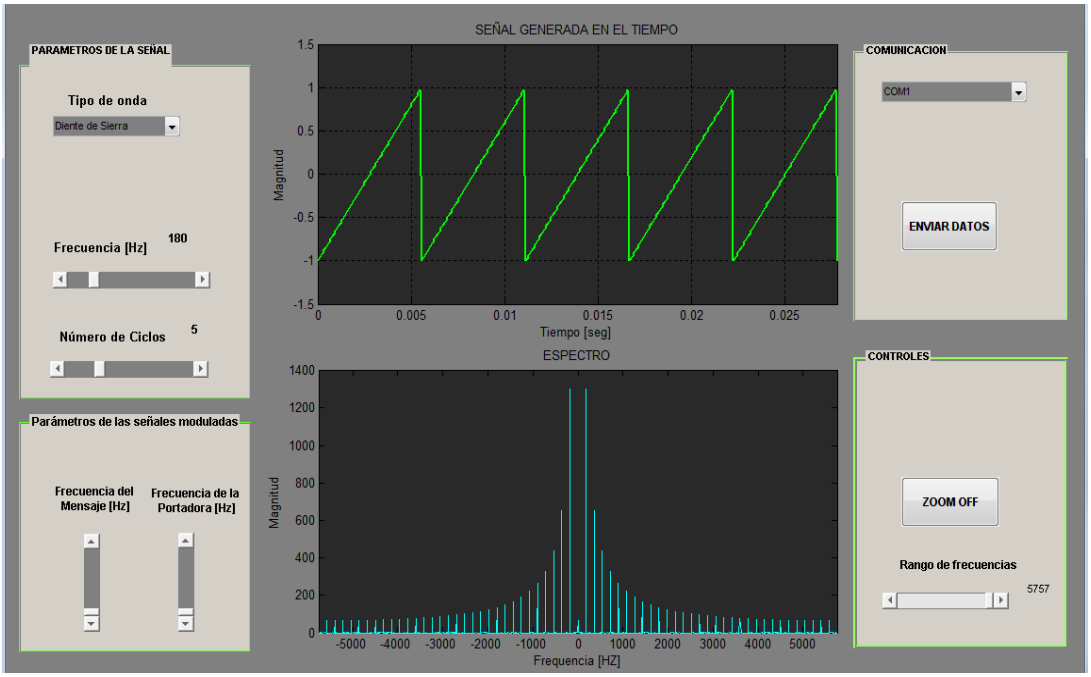


Figura 3-8 Ventana del Generador configurada

El espectro de frecuencia observado en la figura 3-8 es el resultado de dibujar los puntos obtenidos de la función de Matlab *F(Ω) = fft(x(n))*, esta función emplea el algoritmo de la Transformada rápida de Fourier de la señal *x(n)*, donde *x(n)* es una matriz que tiene todos los puntos de la señal en dominio del tiempo discreto de la señal muestreada, que para este caso es la onda periódica “Diente de Sierra”, y *F(Ω)* es la matriz que contiene los datos de la Transformada Discreta de Fourier de la señal *x(n)*.

Figura 3-9 Panel "Parámetros de las Señales Moduladas"

En el Panel **"Parámetros de las Señales Moduladas",** observado en la figura 3-9, se puede configurar las características de las señales moduladas en AM y FM, se ha establecido como señal de mensaje una onda coseno debido a que el alcance de este proyecto tiene fines educativos.

- **Slider "Frecuencia del Mensaje".-** Por medio de este elemento de la interfaz gráfica, podemos configurar la frecuencia de la señal mensaje de las modulaciones AM o FM según se seleccione en el menú Pop – Up "*Tipo de Onda*" del panel "*Parámetros de la Señal*".

En la figura 3-10 se ha seleccionado una frecuencia para la señal mensaje de 40 Hz.

Figura 3-10 Frecuencia de la señal Mensaje

- **Slider "Frecuencia de la Portadora".-** Por medio de este elemento de la interfaz gráfica, podemos configurar la frecuencia de la señal Portadora de las modulaciones AM o FM según se seleccione en el menú Pop – Up "*Tipo de Onda*" del panel "*Parámetros de la Señal*".

En la figura 3-11 se ha seleccionado una frecuencia para la señal portadora de 400 Hz.

Figura 3-11 Frecuencia de la señal Portadora

Las frecuencias a fijar en estos dos sliders está en los rangos de 1 Hz hasta 1000 Hz, por las razones antes mencionadas en la descripción del slider "*Frecuencia [Hz]*" del panel "*Parámetros de la señal*".

En la figura 3-12 observamos la señal en la ventana de gráficos con las características de tipo de onda, frecuencia del mensaje, frecuencia de la Portadora y ciclos de señal configurados en todos los elementos descritos hasta ahora.

Tipo de onda:	*Modulación AM*
Frecuencia:	--
Número de ciclos:	*5*
Frecuencia del Mensaje:	40 Hz
Frecuencia de la Portadora:	400 *Hz*

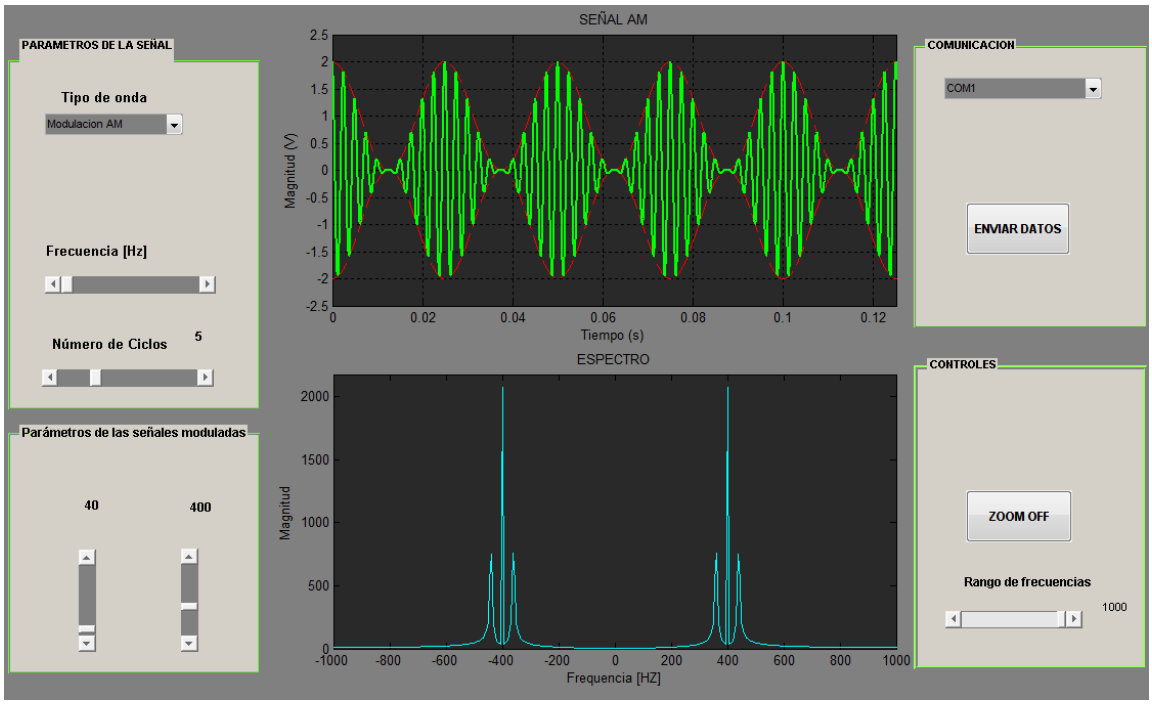


Figura 3-12 Ventana del Generador configurada para señal AM

El espectro de frecuencia observado en la figura 3-12 corresponde al de la señal AM, para este caso específico en el gráfico de la onda en el tiempo se ha resaltado en color rojo la envolvente de la señal AM que corresponde a la onda mensaje de este tipo de modulaciones. También se pueden apreciar los impulsos de frecuencia de la portadora y al lado las bandas laterales de la señal mensaje ubicadas en el gráfico correspondiente al espectro de frecuencia.

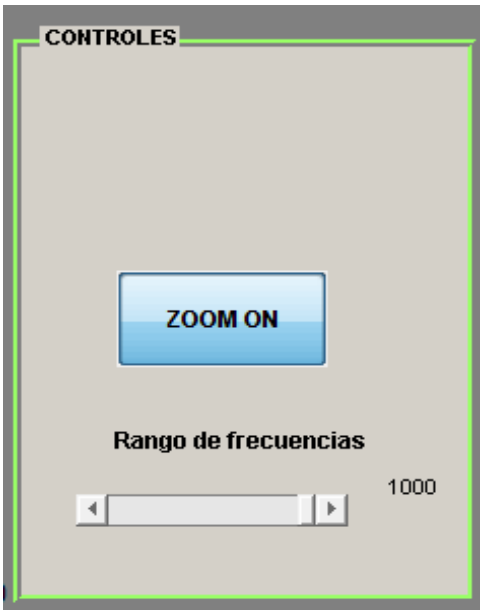


Figura 3-13 Panel "Controles"

El Panel **"Controles"** observado en la figura 3-13 consta de los siguientes elementos:

- **Botón "ZOOM ON".-** Permite realizar un ZOOM en la onda que se seleccione con el puntero del mouse, es decir, se hace un aumento en la apreciación de las ondas seleccionadas con el fin de realizar una inspección minuciosa de las mismas, ya sea en el dominio del tiempo o en el espectro de frecuencia.

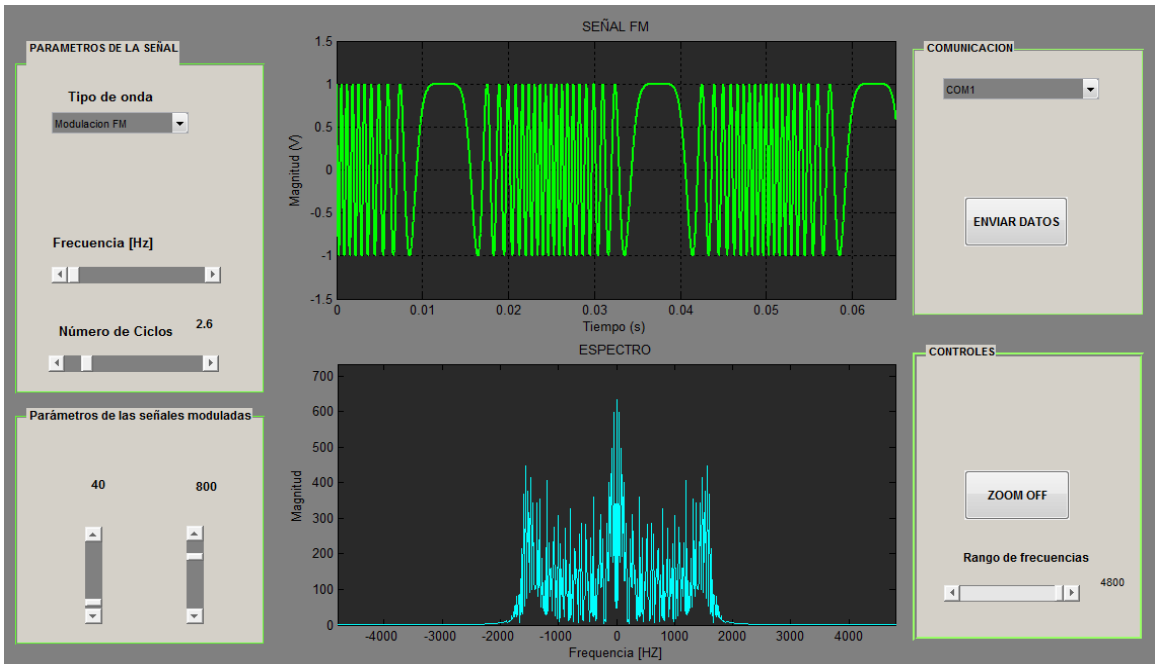


Figura 3-14 Ventana del Generador configurada para señal FM

En la figura 3-14 se ha generado una señal FM con los parámetros configurados de la siguiente manera:

Tipo de onda:	*Modulación FM*
Frecuencia:	--
Número de ciclos:	*2.6*
Frecuencia del Mensaje:	40 Hz
Frecuencia de la Portadora:	800 Hz

Estos parámetros fueron seleccionados con los elementos de interfaz gráfica descritos anteriormente. Al observar el espectro de frecuencias de esta señal nos fijamos que la apreciación de la misma no es muy buena, ya que este espectro es muy estrecho debido a las características de los parámetros configurados. En este caso será necesario hacer una ampliación de la misma para tener una vista más detallada de los impulsos generados por la modulación FM.

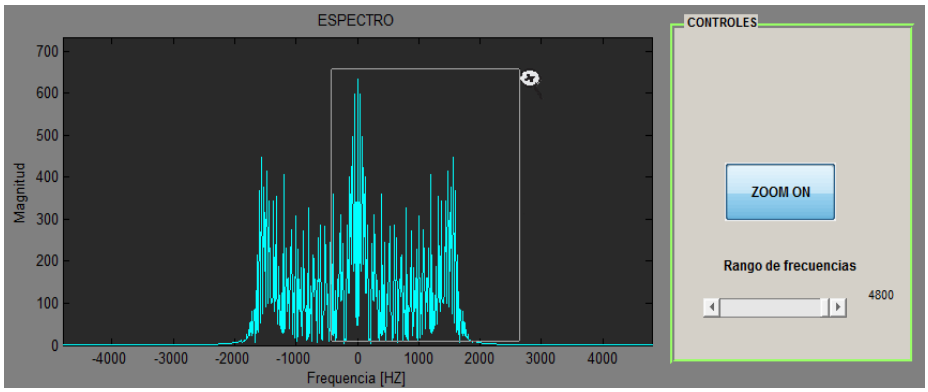


Figura 3-15 Gráfico sin ampliar con el botón “ZOOM ON”

En la figura 3-15 se está procediendo a hacer una ampliación mediante el uso del botón “ZOOM ON”, para esto se necesita seleccionar el área a ser ampliada con el puntero del mouse.

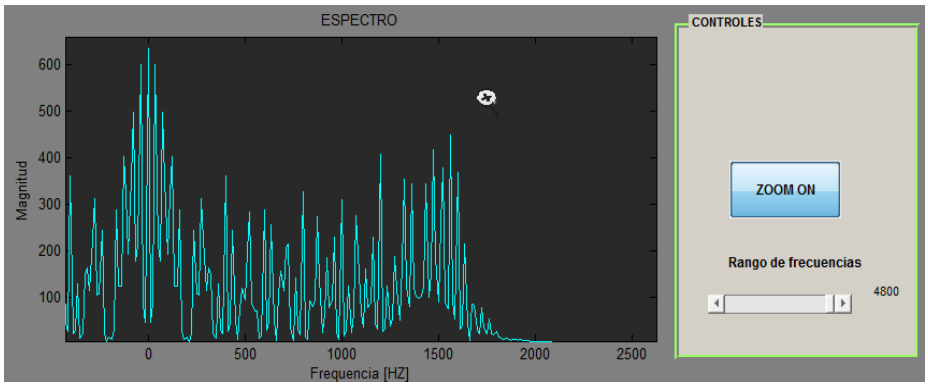


Figura 3-16 Gráfico ampliado con el botón “ZOOM ON”

En la figura 3-16 ya tenemos una gráfica ampliada del espectro de la señal generada anteriormente, aquí se observa de manera más detallada los impulsos generados en las distintas bandas de frecuencias del espectro. Si se quisiera se podría ampliar aún más este gráfico simplemente repitiendo el procedimiento descrito en la figura 3-15.

- **Slider "Rango de Frecuencias".-** Permite configurar la frecuencia inicial y final que se graficaría en el espectro de frecuencias de cualquier señal generada, el valor por defecto de este rango es:

$$Rango\ de\ Frecuencias\ por\ defecto = \frac{2 * \mathrm{f}}{2000}$$

Donde *f* es la frecuencia de la señal en el dominio del tiempo cuando se configura una onda no modulada, y cuando se configura una onda modulada *f* toma el valor de la frecuencia de la portadora usada para la modulación.

Esta es una forma alternativa al botón "ZOOM ON" para hacer ampliación del gráfico, la diferencia es que el botón "ZOOM ON" hace una ampliación en una zona específica del espectro, mientras que el slider "Rango de Frecuencias" redibuja el espectro para un rango de frecuencias configurado por el usuario.

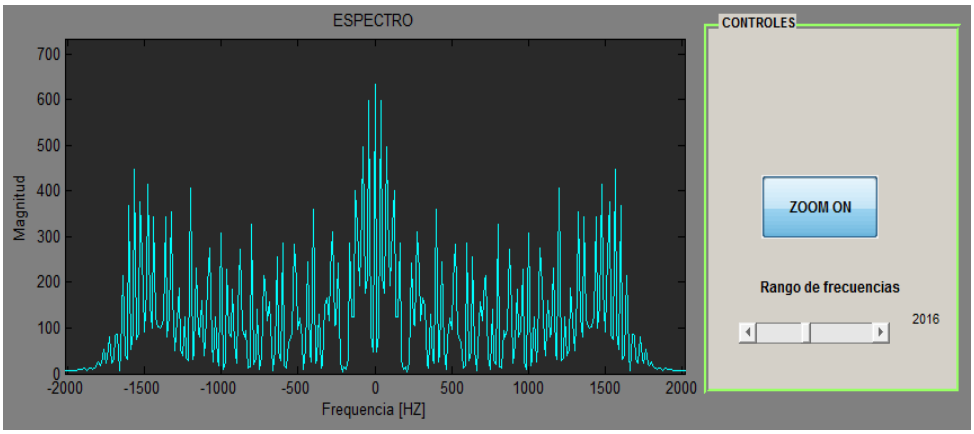


Figura 3-17 Gráfico ampliado con el slider "Rango de Frecuencias"

La figura 3-17 muestra un espectro dibujado en un rango de frecuencias que va desde -2.016 HZ hasta 2016 HZ, el cual fue seleccionado en el slider “Rango de Frecuencias”.

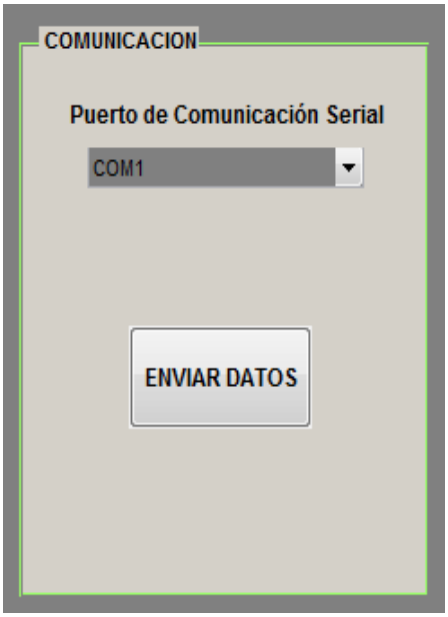


Figura 3-18 Panel “Comunicación”

El Panel **“Comunicación”** observado en la figura 3-18 consta de los siguientes elementos:

- **Menú Pop – up “Puerto de Comunicación Serial”.-** Permite seleccionar el puerto de comunicación entre el computador y la tarjeta DE 2 mediante un cable de conexión serial.

La figura 3-19 muestra la variedad de puertos que el usuario puede seleccionar por medio del puntero del mouse que van desde el puerto “COM1” hasta el puerto “COM12”, este rango se ha programado teniendo en cuenta que no sabemos cuántos puertos hay disponibles en el computador en el que se vaya a hacer uso del generador de funciones desarrollado en este proyecto.

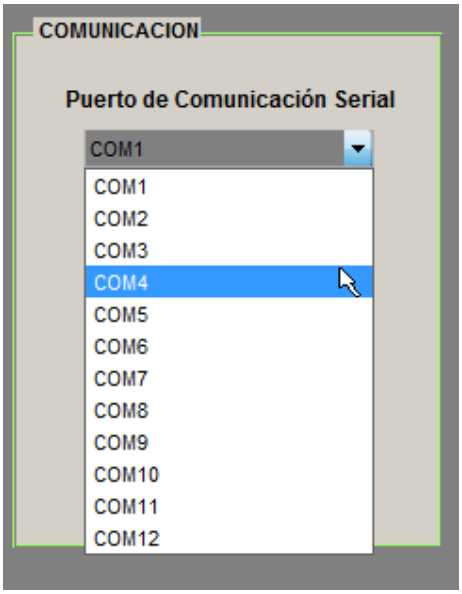


Figura 3-19 Puerto de Comunicación Serial

El puerto a elegir debe ser verificado previamente para evitar errores en la comunicación serial, ya que esta selección la hace automáticamente el computador y depende de que puertos de comunicación serial tenga disponibles al momento de la conexión del cable al puerto RS232.

- **Botón "ENVIAR DATOS".-** Activa la transmisión serial de la onda generada, es decir, envía datos muestreados desde el computador hacia la tarjeta DE 2.

Al presionar el botón *"ENVIAR DATOS"* (ver figura 3-18), el generador de funciones ejecuta códigos de envío de datos a través del puerto serial hacia la tarjeta. A continuación se muestra parte del código del Generador de Funciones.

```
%--- Código para la Comunicación Serial ---%

%--- Parametrización del puerto de Comunicación Serial
    SerDE2 = serial(puerto);
set(SerDE2,'BaudRate',115200);
set(SerDE2,'DataBits',8);
set(SerDE2,'Parity','odd');
set(SerDE2,'StopBits',1);
set(SerDE2,'FlowControl','none');
fopen(SerDE2);
%*-*-*-*-*-*-*
```

Figura 3-20 Código en Matlab para configurar el Puerto Serial

En el código mostrado en la figura 3-20 contiene la información de cómo se ha configurado el puerto serial desde Matlab, es decir, esta es la configuración del Generador de Funciones para la comunicación entre el computador y la Tarjeta DE2.

```
%--- Código PCM para la señal muestreada, con una resolución de 12 Bits ---%
for k = 1:1024 %Selecciona los primeros 1024 datos de muestra generados
    j = 1;
    g = 0;
%--- Este lazo "for" se lo usa para llevar los primeros 1024 de los 4096 mues-
%treados generados anteriormente a rangos de amplitud de 0 a 4095, datos de 12bits
    for j = 1:4096
        if (((j-1)/4096) <= y(k) <= (j/4096))
            dato(k) = y(k) * 4095;
            g = 1;
        end
        if g == 0
            continue
        else
            break
        end
    end
datoent(k) = round(dato(k));%Redondea las Muestras haciendolas Números enteros
datobin = dec2bin(datoent(k),12);%Comvierte los datos de Decimal a Binario de 12 Bits
%--- Se envian los 5 bits más significativos
datoh(k) = bin2dec(datobin(1:5));
if (datoh(k) == 0)%Se codifica el Número "0" como un Caracter especial: "160"
%--- Se envian los datos Codificados por el puerto de Comunicación Serial
        fprintf(SerDE2,'%c', char(160));
    else
        fprintf(SerDE2,'%c', char(datoh(k)));
    end
    datol(k) = bin2dec(datobin(6:12));
%--- Se envian los 12 bits menos significativos
if (datol(k) == 0)%Se codifica el Número "0" como un Caracter especial: "160"
%--- Se envian los datos Codificados por el puerto de Comunicación Serial
    fprintf(SerDE2,'%c', char(160));
    else
        fprintf(SerDE2,'%c', char(datol(k)));
    end
end
%*-*-*-*-*-*
```

Figura 3-21 Código en Matlab para codificación PCM y envío de datos por el Puerto Serial

Las líneas de código observadas en la figura 3-21 son las usadas para la codificación PCM y envío de los datos muestreados por el Generador de Funciones. Se ha hecho una codificación PCM de 12 bits de

resolución con 1024 muestras que van desde valores de 0 hasta 4095. Estos 12 bits de valores PCM se lo ha dividido en dos paquetes de datos de un byte cada uno, el primer byte enviado corresponde a los 5 bits más significativos de cada muestra generada, mientras que el segundo byte enviado contiene los siguientes 7 bits menos significativos de la muestra. Este procedimiento se ejecuta para las 1024 muestras a ser enviadas.

```
%--- Se Cierra el puerto COM al finalizar ---%
fclose(SerDE2);
delete(SerDE2)
clear SerDE2
disp('STOP')
%*-*-*-*-*-*-
```

Figura 3-22 Código en Matlab para cerrar el puerto Serial

Para cerrar el puerto que se ha utilizado en el computador, se usa las líneas mostradas en la figura 3-22, este procedimiento es necesario y se realiza con el fin de evitar problemas con el funcionamiento de los puertos de comunicación usados en futuros envios de datos.

3.3 Máquina desarrollada en la tarjeta DE2

Con el fin de realizar una explicación detallada del hardware diseñado en la tarjeta DE2, los párrafos siguientes se encargan de despejar cualquier duda sobre los distintos elementos configurados en la herramienta de diseño de hardware de Altera SOPC Builder, los cuales son esenciales y básicos en cualquier máquina digital.

3.3.1 Diseño del Hardware en SOPC Builder

Para implementar sistemas basados en el procesador Nios II se usa el Software de Altera SOPC Builder. Cualquier sistema básico requiere de componentes funcionales tales como memorias, puertos de entrada/salida, interfaces de comunicación, etc.

SOPC Builder es la herramienta de software que nos facilita el diseño e implementación de sistemas destinados a ser cargados en chips programables.

Las partes de nuestro sistema desarrollado en SOPC Builder son:

- **Procesador Nios II.-** Unidad de Procesador Central (CPU).
- **JTAG UART.-** Interfaz de comunicación con el computador.
- **RS232 UART.-** Puerto para la comunicación serial entre el computador y la tarjeta DE2 de Altera.

- **VGA Controller.-** Controlador que se encarga del manejo de los diferentes parámetros de la interfaz VGA.

En la figura 3-23 se puede observar las diferentes componentes de hardware de nuestro sistema, tomados de la máquina media de University Program de Altera que proporciona soporte completo para introducir a los estudiantes a la tecnología digital. El apoyo incluye hardware, software y materiales de enseñanza. [16]

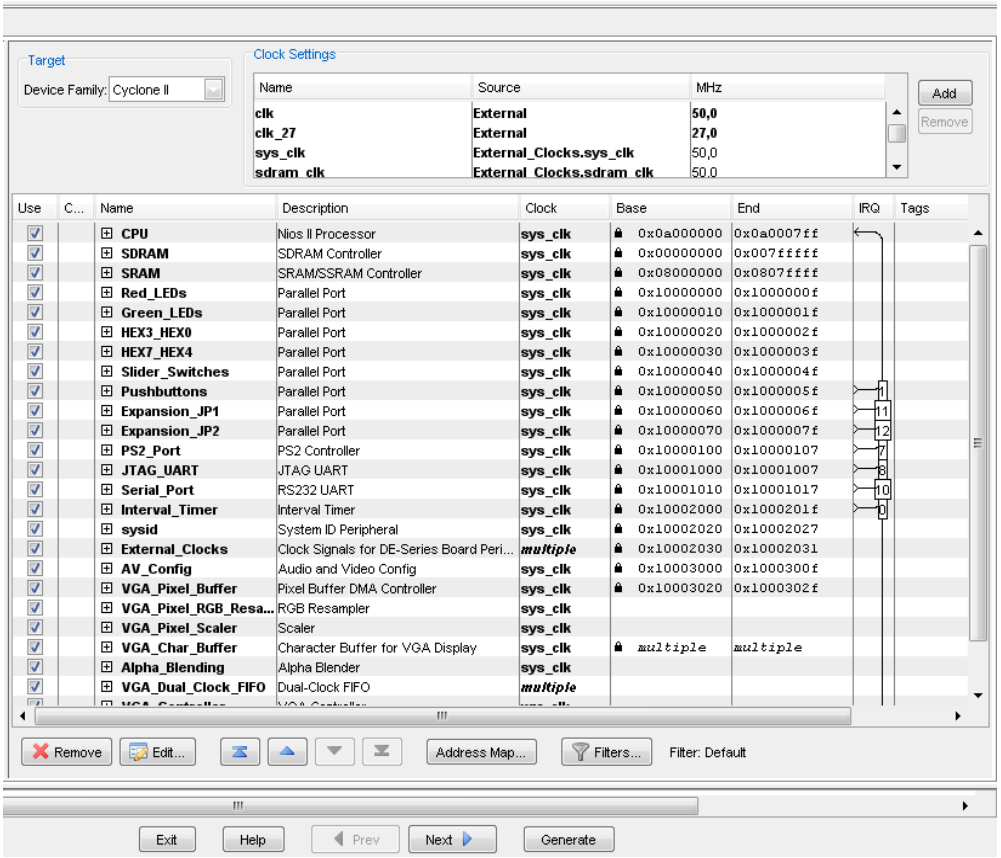


Figura 3-23 Sistema en SOPC Builder

3.3.2 RS232 UART

La principal característica de este tipo de comunicación es que los datos se envían uno después del otro, por lo cual se denomina comunicación serial, a diferencia de la comunicación en paralelo que permite enviar un grupo de datos simultáneamente. El protocolo de comunicación serial utilizado en el proyecto es el RS232. Se escogió este protocolo de comunicación debido a que la tarjeta DE2 cuenta con el puerto necesario.

Como se muestra en la figura 3-24, los parámetros escogidos para el controlador RS232 son los siguientes:

- **Baud Rate.-** La velocidad de baudios escogida es de 115200 bps debido a que se requiere de la máxima velocidad de transmisión serial de datos para evitar retrasos ocasionados por la gran cantidad de muestras que se envían a la tarjeta.
- **Parity.**- La paridad escogida es “Odd”, que es la misma configurada en el software Generador de Funciones desarrollada en MATLAB, y se la usa para la detección de errores en la comunicación serial de datos.

- **Data Bits.-** La longitud de los datos elegida es de ocho bits, ya que los datos enviados desde el software generador de funciones se encuentran codificados en PCM en muestras de tamaño de ocho bits.
- **Stop Bits.-** Se mantiene un bit de parada, tal como viene la configuración por defecto.

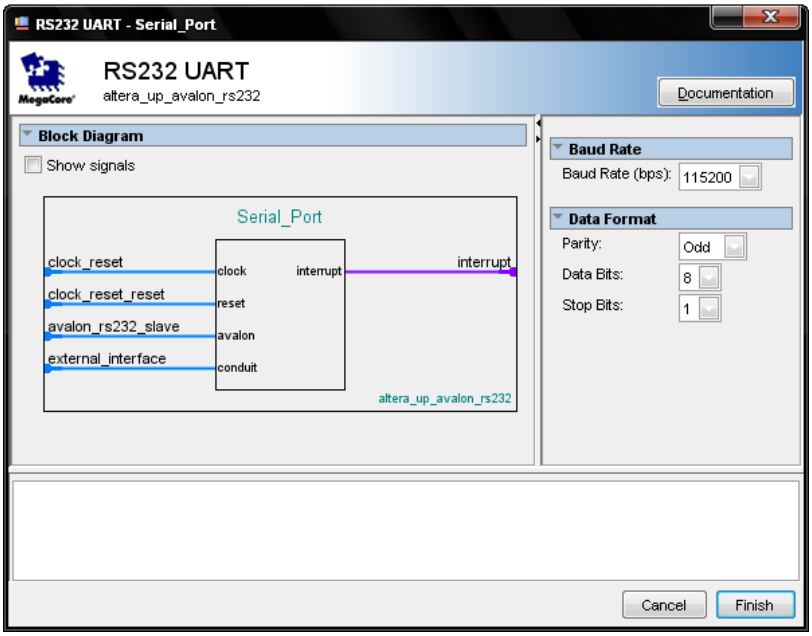


Figura 3-24 Configuración del puerto RS232 en SOPC Builder

3.3.3 VGA Controller

VGA (Video Graphics Array) es un estándar de pantallas de video. El controlador VGA genera la sincronización de señales y salidas seriales de datos de pixeles. La implementación en SOPC Builder del controlador se muestra en la figura 3-25.

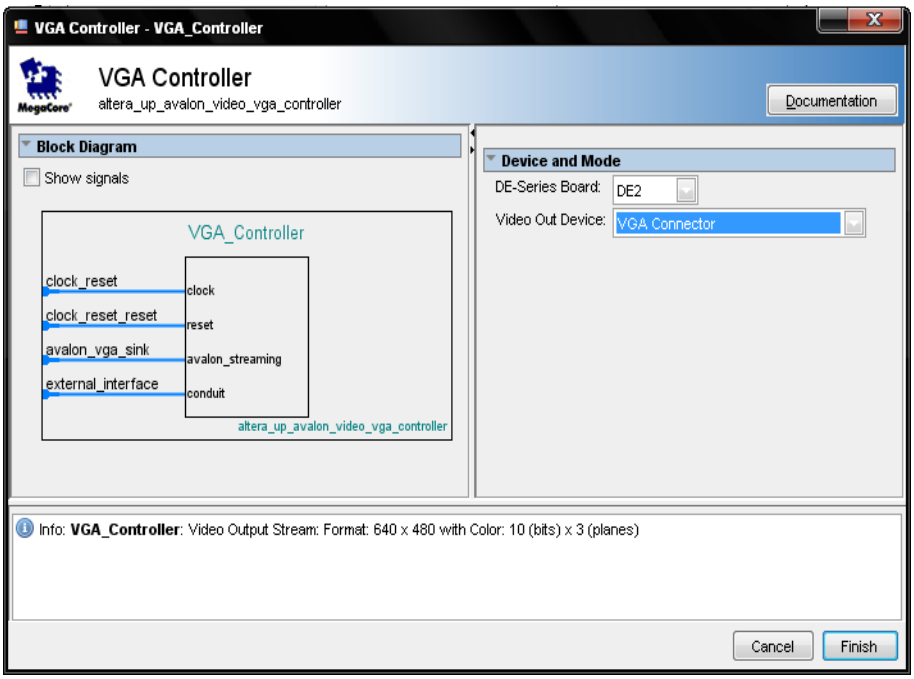


Figura 3-25 Controlador VGA

La figura 3-26 muestra algunos parámetros de configuración necesarios para el controlador VGA. Vemos que la resolución utilizada es de 320 x 240 debido a que esta es la máxima resolución soportada por el controlador usado en este proyecto. El factor escalador utilizado es de 2, para obtener la impresión de resolución de 640 x 480, es decir que se

multiplican el ancho y alto de los pixeles con el fin de aparentar una mayor resolución de pantalla. El color de cada pixel se representa por medio de un número de 16 bits, en donde los 5 primeros (bits del 0 al 4) representan el tono azul, los seis siguientes (bits del 5 al 10) representan el tono verde y los cinco últimos (bits del 11 al 15) representan el tono rojo del pixel.

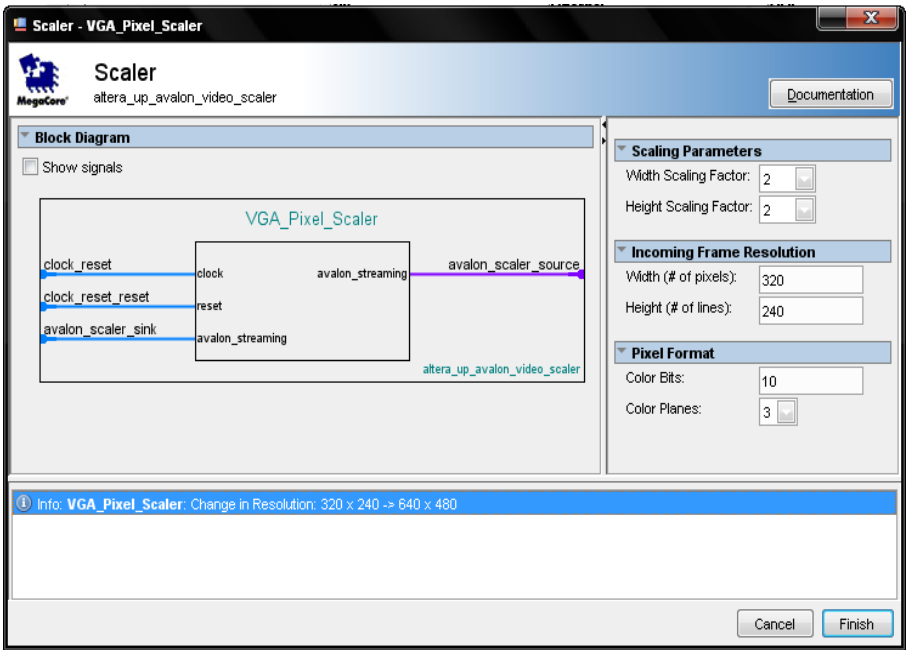


Figura 3-26 VGA Pixel Scaler

3.4 Procesamiento y presentación de los datos en un Monitor VGA

En esta etapa se hace la presentación de los datos procesados con el algoritmo FFT implementado en el procesador NIOS II de la tarjeta.

3.4.1 Programación en lenguaje C utilizando NIOS II

Para el cálculo de la transformada rápida de Fourier se implementó un código en lenguaje C que se describe a continuación.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define rs232bytes_available(c) (c>>16)

volatile unsigned int *pRS232 = 0x10001010;

void write_pixel (int, int, short);
void VGA_text (int, int, char *);
void VGA_box (int, int, int, int, short);
void fft (int n, int m, double x[], double y[]);
void magnitud (int n, double x[], double y[], double r[]);
void entero (int n, double m[], int r[]);
int position (int n, int m[]);
int maximo (int n, int m[]);
int getcharRS232();
void fft2 (int n, int m, double x[], double y[]);
```

Figura 3-27 Declaración de librerías, punteros y funciones

La figura 3-27 muestra una fracción del código FFT implementado, donde podemos apreciar las librerías que se utilizan, los punteros a utilizar y las funciones necesarias para el desarrollo del proyecto.

LIBRERÍAS

Las librerías utilizadas en el código implementado en lenguaje C se muestran en la Figura 3-27 y son las básicas y necesarias para cualquier código en lenguaje C.

Librería stdio.h.- “Standard input – output header”, librería estándar de lenguaje C que contiene las definiciones de macros, constantes, declaraciones de funciones y definición de tipos utilizados por varias operaciones estándar de entrada y salida.

Librería stdlib.h.- “Standard library”, librería estándar de propósito general de lenguaje C. Contiene los prototipos de funciones de C para gestión de memoria dinámica y control de procesos. Esta librería es necesaria para poder realizar conversiones de tipo de dato, procesos de orden y búsqueda y ciertas operaciones matemáticas como valor absoluto y divisiones enteras.

Librería math.h.- Archivo de cabecera de la librería estándar de lenguaje C diseñada para las operaciones matemáticas básicas, potencia de un número, logaritmo, funciones trigonométricas, etc.

PUNTEROS

El único puntero utilizado en el algoritmo FFT implementado en lenguaje C es el *pRS232* que apunta a la dirección de memoria 0x10001010 del procesador NIOS. Es en esta dirección de memoria en la que se almacenan los datos recibidos por el puerto serial de la tarjeta DE2, que en nuestro caso corresponden a los datos muestreados de la señal en el tiempo escogida por el usuario a través de la interfaz gráfica realizada en MATLAB y enviadas por medio de un cable USB – Serial desde el computador a la tarjeta DE2. Los datos recibidos son leídos en el código implementado haciendo uso del puntero *pRS232*. La constante definida como *rs232bytes_available* es importante en la lectura del buffer del RS232 de la tarjeta. Su función en el código se explica más adelante.

FUNCIONES

La Figura 3-27 muestra las funciones creadas, necesarias para lograr el fin del proyecto, calcular y mostrar la transformada de Fourier en un monitor VGA utilizando el algoritmo de la transformada rápida de Fourier.

```
/*************************************************************
 * Get a character from the RS232 UART
*************************************************************
int getcharRS232()
{
        unsigned int rs232word;

        do
                rs232word = *pRS232;
        while( rs232bytes_available(rs232word) == 0 );

        return rs232word & 0xFF;
}

/*************************************************************
```

Figura 3-28 Función int getcharRS232( )

Función getcharRS232

La Figura 3-28 muestra el código de la función *getcharRS232*. Esta función se encarga de verificar si hay datos en el buffer del RS232 de la tarjeta y pasar esos datos a otra variable. Como observamos en la figura, primero crea una variable entera sin signo rs232word. Luego ingresa a un *do – while* en donde primero asigna a *rs232word* el contenido del puntero *pRS232*, es decir los datos recibidos desde el computador. En la condición del *while* vemos que utiliza la constante *rs232bytes_available* para mover el puntero hacia la dirección del buffer y verificar si es que existen bytes disponibles, es decir, si hay datos sin leer en el buffer del RS232 de la tarjeta. Esta función no sale del lazo *do – while* hasta haber recibido un dato. Una vez que se ha verificado la existencia de información en el buffer se retorna este dato en *rs232word*.

```
/*************************************************************
 * Integer transform and rounded
*************************************************************
void entero (int n, double m[], int r[])
{
        int i, temp;
        double t;

        for (i = 0; i < n; i++)
        {
                t = m[i]*10;
                temp = (int) t % 10;
                if (temp >= 5)
                        r[i] = ((int) m[i]) + 1;
                else
                        r[i] = (int) m[i];
        }
}

/*************************************************************
```

Figura 3-29 Función void entero (int n, double m[ ], int r[ ])

Función entero

La Figura 3-29 muestra el código de la función *entero*, que se encarga de transformar los datos de un arreglo de tipo flotante a datos de tipo entero. La función recibe como parámetros el número de datos del arreglo, así como el arreglo de flotantes y un arreglo entero en donde se almacenarán los datos transformados.

```
void fft2 (int n, int m, double x[], double y[])
{
        long i, i1, j, k, i2, l, l1, l2;
        double c1, c2, tx, ty, t1, t2, u1, u2, z;

        /* Do the bit reversal */
        i2 = n >> 1;
        j = 0;
        for (i = 0; i < n-1; i++)
        {
                if (i < j)
                {
                        tx = x[i];
                        ty = y[i];
                        x[i] = x[j];
                        y[i] = y[j];
                        x[j] = tx;
                        y[j] = ty;
                }
                k = i2;
                while (k <= j)
                {
                        j -= k;
                        k >>= 1;
                }
                j += k;
        }

        /* Compute the FFT */
        c1 = -1.0;
        c2 = 0.0;
        l2 = 1;
        for (l = 0; l < m; l++)
        {
                l1 = l2;
                l2 <<= 1;
                u1 = 1.0;
                u2 = 0.0;
                for (j = 0; j < l1; j++)
                {
                        for (i = j; i < n; i+=l2)
                        {
                                i1 = i + l1;
                                t1 = u1 * x[i1] - u2 * y[i1];
                                t2 = u1 * y[i1] + u2 * x[i1];
                                x[i1] = x[i] - t1;
                                y[i1] = y[i] - t2;
                                x[i] += t1;
                                y[i] += t2;
                        }
                        z =  u1 * c1 - u2 * c2;
                        u2 = u1 * c2 + u2 * c1;
                        u1 = z;
                }
                c2 = sqrt((1.0 - c1) / 2.0);
                c2 = -c2;
                c1 = sqrt((1.0 + c1) / 2.0);
        }

}
```

Figura 3-30 Función void fft2 (int n, int m, double x[ ], double y[ ])

La variable *i* es utilizada en el lazo *for*, mientras que las variables *t* y *temp* se utilizan para hacer el redondeo correspondiente en caso de que el número flotante se acerque más a un número entero mayor. Por ejemplo, si el número flotante es el 1.7 entonces en *t* tendríamos 17 y

temp sería 7. Como *temp* es mayor que 5, el arreglo entero se carga con 2. Es necesario realizar este artificio debido a que int transforma el flotante a entero pero no realiza el redondeo, en este caso el 1.7 se habría convertido en 1.

Función fft2

La Figura 3-30 muestra la función *fft2* encargada de calcular la transformada rápida de Fourier. Los parámetros que recibe son el número de datos *n*, el exponente de base dos del número de datos *m*, el arreglo en donde se encuentran los datos muestreados de la señal en el tiempo, que además será el arreglo en el que se almacenará la parte real de la transformada de Fourier *x[ ]* y el arreglo en donde se almacena la parte imaginaria de la transformada de Fourier *y[ ]*. Es importante tener en cuenta que esta función puede calcular la transformada de datos complejos, en este caso tanto la matriz *x* como la matriz *y* estarán con datos inicialmente, que luego serán reemplazados por la parte real e imaginaria de la transformada. En nuestro caso solo utilizamos datos reales, por lo que es necesario encerar la matriz imaginaria antes de utilizar la función para evitar errores en el cálculo.

```
/*****************************************************************************
 * Draw a filled rectangle on the VGA monitor
*****************************************************************************/
void VGA_box(int x1, int y1, int x2, int y2, short pixel_color)
{
        int offset, row, col;
        volatile short * pixel_buffer = (short *) 0x08000000;    // VGA pixel buffer

        /* assume that the box coordinates are valid */
        for (row = y1; row <= y2; row++)
        {
                col = x1;
                while (col <= x2)
                {
                        offset = (row << 9) + col;
                        *(pixel_buffer + offset) = pixel_color; // compute halfword address, set pixel
                        ++col;
                }
        }
}

/*****************************************************************************
```

Figura 3-31 Función void VGA_box

Función VGA_box

La Figura 3-31 muestra el código de la función *VGA_box*. Esta función se encarga de dibujar un recuadro en el monitor VGA, recibe como parámetros la primera posición en el eje x, *x1*, la primera posición en el eje y, *y1*, la segunda posición en el eje x, *x2*, la segunda posición en el eje y, *y2,* y el color de los pixeles (*pixel_color*). Así la función puede dibujar el recuadro desde unas coordenadas de inicio hasta las coordenadas de fin, indicando el color de los pixeles.

```
/*****************************************************************************
 * Subroutine to send a string of text to the VGA monitor
*****************************************************************************/
void VGA_text(int x, int y, char * text_ptr)
{
        int offset;
        volatile char * character_buffer = (char *) 0x09000000; // VGA character buffer

        /* assume that the text string fits on one line */
        offset = (y << 7) + x;
        while ( *(text_ptr) )
        {
                *(character_buffer + offset) = *(text_ptr);     // write to the character buffer
                ++text_ptr;
                ++offset;
        }
}

/*****************************************************************************
```

Figura 3-32 Función void VGA_text (int x, int y, char * text_ptr)

Función VGA_text

La Figura 3-32 muestra el código de la subrutina *VGA_text*, utilizada para escribir caracteres en el monitor VGA. La cadena debe pasarse al buffer de caracteres del VGA para que esta pueda ser mostrada en el monitor. Cada carácter dibujado en pantalla ocupa 4 pixeles tanto en el eje x como en el eje y. Los parámetros necesarios para la función son posición en x (*int x*), posición en y (*int y*) y la cadena a ser dibujada *text_ptr*. Es importante que esta cadena termine con el carácter *'\0'*, correspondiente al fin de cadena.

```
/*****************************************************************************
 * Subroutine to write a pixel
*****************************************************************************/
void write_pixel (int x, int y, short colour)
{
        volatile short *vga_addr=(volatile short*)(0x08000000 + (y<<10) + (x<<1));
        *vga_addr=colour;
}

/*****************************************************************************
```

Figura 3-33 Función void write_pixel (int x, int y, short colour)

Función write_pixel

En la Figura 3-33 se puede observar el código de la función *write_pixel*. Esta función nos permite escribir un solo pixel en el monitor VGA. Los parámetros necesarios son la posición en el eje x (*int x*), la posición en el eje y (*int y*) y el color del pixel *short colour*.

```
/*****************************************************************
 * Magnitude
*****************************************************************
void magnitud (int n, double x[], double y[], double r[])
{
        int i;
        double n1, n2;

        for (i = 0; i < n; i++)
        {
                n1 = pow(x[i],2);
                n2 = pow(y[i],2);
                r[i] = sqrt(n1 + n2);
        }
}

/*****************************************************************
```

Figura 3-34 Función void magnitud

Función magnitud

La Figura 3-34 muestra el código de la función magnitud. Esta función se encarga de llenar un arreglo con la magnitud de los números complejos almacenados en dos arreglos que contienen la parte real e imaginaria. Los parámetros que necesita la función son el número de elementos en los arreglos *n*, la matriz correspondiente a la parte real de los números complejos *x[ ]*, la matriz correspondiente a la parte

imaginaria de los números complejos *y[ ]* y finalmente el arreglo en el cual se almacena la magnitud *r[ ]*. Debido a que el proyecto consiste en graficar la magnitud de la transformada de Fourier, esta función es de gran utilidad. En la Figura 3-34 vemos que el código de la función es realmente sencillo. Simplemente se ingresa en un lazo *for* y se calcula el cuadrado tanto de la parte real como imaginaria para luego guardar la raíz de ambos cuadrados en el arreglo de la magnitud.

```
/*************************************
 * Max of an array and his position
*************************************
int position (int n, int m[])
{
        int i, pos, max = 0;
        for (i = 0; i < n; i++)
        {
                if (m[i] > max)
                {
                        max = m[i];
                        pos = i;
                }
        }
        return pos;
}
```

Figura 3-35 Función int position (int n, int m[ ])

Función position

El código que se observa en la Figura 3-35 corresponde al de la función *position*, la cual obtiene la posición del mayor número de un arreglo de enteros. Los parámetros que se necesitan en esta función son el número de elementos del arreglo *int n* y el arreglo de enteros *m[ ]*. El código de la función también resulta ser muy sencillo y básico.

Simplemente se ingresa a un lazo *for* en el que se compara cada número del arreglo con la variable *max* que inicialmente vale cero. Si el número del arreglo es mayor que *max* entonces ese número se pasa a *max* y la posición de ese número se almacena en la variable *pos*.

```
int maximo (int n, int m[])
{
        int i, max = 0;
        for (i = 0; i < n; i++)
        {
                if (m[i] > max)
                        max = m[i];
        }
        return max;
}

/*************************************
```

Figura 3-36 Función int maximo (int n, int m[ ])

Función máximo

En la Figura 3-36 se encuentra el código correspondiente a la función *máximo*, la misma que, como indica su nombre, obtiene el mayor número de un arreglo de enteros. Los parámetros que se necesitan en esta función son el número de elementos del arreglo *int n* y el arreglo de enteros *m[ ]*. El código de la función también resulta ser muy sencillo y básico. Simplemente se ingresa a un lazo *for* en el que se compara cada número del arreglo con la variable *max* que inicialmente vale cero. Si el número del arreglo es mayor que *max* entonces ese número se pasa a *max*.

3.4.2 Código del Proyecto

En esta sección se explica paso a paso lo que se ha realizado en el código del proyecto, en donde se utilizan las funciones ya descritas.

```
int main(void)
{
        int i, j, xp, n, exp, max, pm, esc, v, yp, pi, pf, t;
        int u, d, c, m, mm, var, fme, b = 0, pot10, cant;
        double df, f;
        double frec[8];
        double real[1024], imag[1024], sig[1024];
        int rmag[1024], rsig[1024];
        char cero[3] = "0\0";
        char uno[3] = "1\0";
        char muno[4] = "-1\0";
        char limp[4] = "  \0";
        char limpia[16] = "               \0";
        char doscientos[] = "200\0";
        char ey[5], exp10[6];
```

Figura 3-37 Variables locales

La Figura 3-37 muestra las variables locales, es decir las que se utilizan dentro del programa principal. La función de cada una de ellas se definirá conforme avancemos en la explicación del algoritmo principal.

```
VGA_box (0, 0, 319, 239, 0x0000);

while (1)
{
        //encerar matriz imaginaria para evitar errores
        for (i = 0; i < 1024; i++)
                imag[i] = 0;

        j = 0; i = 0; xp = 29; yp = 141; esc = 1;
        n = 1024; exp = 10;
        pi = 0; pf = 129;

        //recepcion de datos codificados
        while (i < n)
        {
                v = getcharRS232();
                if (v == 160)
                        t = 0;
                else
                        t = (v << 7);

                v = getcharRS232();
                if (v == 160)
                        real[i] = t | 0;
                else
                        real[i] = t | v;

                i++;
        }
```

Figura 3-38 Fracción de código 1

En la Figura 3-38 se empieza con el programa propiamente. Primero utilizamos la función *VGA_box* para dibujar en toda la pantalla un recuadro de color negro, es decir, pintamos la pantalla entera con pixeles negros. Podemos apreciar que la posición inicial tanto en el eje x como en el eje y es cero, la posición final del eje x es 319 y la posición final del eje y es 239, ocupando así toda la pantalla de resolución 320 x 240. El color de los pixeles es 0x0000, equivalente a negro.

Luego ingresamos en un lazo *while* infinito, ya que las operaciones descritas en este código deben realizarse de manera indefinida.

Procedemos a encerar el arreglo *imag* correspondiente a la matriz de datos imaginarios. Esta acción se realiza para evitar errores al momento de calcular la transformada de Fourier con la función *fft2*.

Se inicializan algunas variables como *i* y *j*, las cuales se enceran, estas variables son utilizadas en los lazos *for* al trabajar con arreglos. Otra de las variables que se inicializa es *n* que contiene el número de datos que se envían desde el *computador*. Esta cantidad es fija e igual a 1024. La variable *exp* también se inicializa con el exponente de base dos de la cantidad de datos, en este caso *exp* = 10, ya que 2 elevado a la potencia 10 equivale a 1024.

A continuación en el código, se espera a que se envíen datos serialmente desde el computador. Vemos como se ingresa en un lazo *while* que espera a que se recepten los 1024 datos. Podemos ver que cada vez que la variable *i* incrementa en uno, en realidad se han recibido dos datos. Esto se debe a la codificación previa realizada en MATLAB. Un número se envía en dos partes, primero la parte alta y luego la baja. Estos dos datos finalmente son concatenados y almacenados en el arreglo *real[ ]*. La función utilizada para obtener el dato desde el buffer del RS232 de la tarjeta es *getcharRS232()*, la misma que se llama dos veces para obtener la parte alta y baja del número, lo que se traduce en 2048 datos recibidos.

```
//decodificacion de datos
for (i = 0; i < n; i++)
        real[i] = 2 * ((real[i]/4095) - 0.5);

printf("DATOS RECIBIDOS\n");

//adecuacion de datos para ser graficados (161)
for (i = 0; i < 1024; i++)
        sig[i] = real[i] * -32;
entero (1024, sig, rsig);

df = 64 * (float)fme / (float)n ;       //diferencial de frecuencia

for (i = 1; i <= 8; i++)
        frec[i - 1] = i * 16 * df;

pot10 = 0;
while (frec[7] > 10)
{
        frec[7] = frec[7] / 10;
        pot10++;
}

for (i = 0; i < 7; i++)
        frec[i] = frec[i] / pow(10, pot10);

//funcion para calcular la FFT
fft2 (n, exp, real, imag);
printf("FFT CALCULADA\n");
```

Figura 3-39 Fracción de código 2

La Figura 3-39 continúa con el código principal. Aquí vemos el siguiente paso luego de la recepción de los 2048 datos enviados (1024 datos enviando parte alta y baja por separado). Se procede a realizar una decodificación de la codificación hecha en MATLAB. Los datos enviados desde la interfaz gráfica se codifican asignando una escala del 0 al 4095 según el valor real del dato. En el código principal simplemente se hace el proceso inverso, dividiendo los datos para 4095, lo cual nos deja datos entre 0 y 1. Luego restamos 0.5 para tener datos entre -0.5 y 0.5 y finalmente multiplicamos por 2 para obtener datos entre -1 y 1.

Ahora debemos adecuar estos datos para que puedan ser graficados correctamente en el monitor VGA. Para ello multiplicamos cada dato por

32, lo que nos deja con datos entre -32 y 32 en la matriz *sig[ ]*. Estos datos aún son flotantes, por lo que utilizamos la función entero para llenar *rsig[ ]* con el entero de los datos del arreglo *sig[ ]*. El arreglo *rsig[ ]* es el que se utiliza para graficar la señal en el tiempo.

A continuación calculamos el diferencial de frecuencia *df*, este valor es de vital importancia para poder saber la frecuencia de la señal que ha sido enviada.

El arreglo *frec[ ]* es utilizado para almacenar las frecuencias que se muestran en la escala de frecuencia de la gráfica en el monitor VGA.

3.5 Montaje final de los elementos necesarios para la puesta en marcha de este Proyecto.

Para un desarrollo exitoso de las pruebas que se realizan en el capítulo 4 se ha procedido a ubicar cada uno de los elementos como se muestra en la figura 3-40, en esta se puede observar que se utilizan dos monitores, uno que muestra las señales generadas por MATLAB y el monitor utilizado para mostrar el resultado final de los datos procesados por la tarjeta DE2, conectado a la misma mediante un cable VGA. El

cable serial (DB9 – USB) sirve de comunicación entre la tarjeta y el software desarrollado en MATLAB.

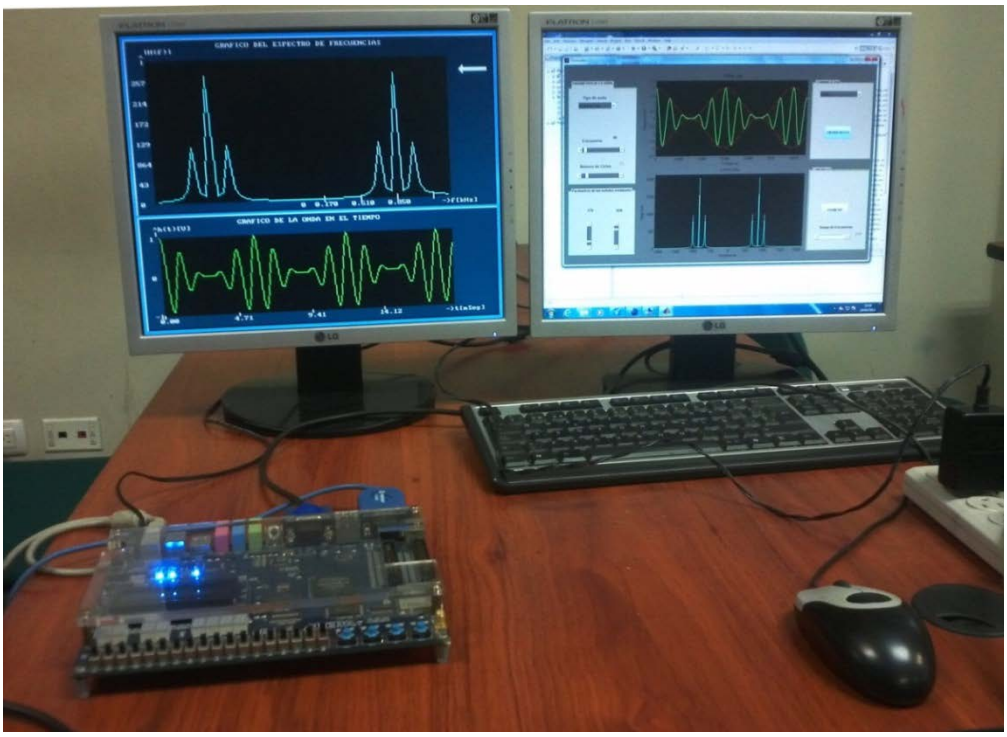


Figura 3-40 Ubicación de los elementos del proyecto

CAPITULO 4

4. PRUEBAS Y RESULTADOS

En este capítulo se muestra las pruebas realizadas, desde el proceso de generación de la onda que se va a procesar, la transmisión serial de los datos generados desde el computador hacia la tarjeta DE2, hasta la presentación de los resultados en un monitor VGA. Para una apreciación detallada del proceso global de este proyecto se ha incluidos cuatro escenarios importantes descritos a continuación.

4.1 Escenarios

- Escenario A: Envío fallido de la onda generada debido a la selección incorrecta del puerto de comunicación del computador.

- Escenario B: Envío exitoso de la onda generada debido a la selección correcta del puerto de comunicación del computador.

- Escenario C: Envío fallido de una nueva onda generada debido a la selección incorrecta del puerto de comunicación del computador, si previamente se obtuvo éxito en lo planteado en el escenario B.

- Escenario D: Envío exitoso de una onda generada y comparación numérica de los resultados mostrados en la pantalla del monitor VGA.

- Escenario E: Pruebas de rendimiento del procesador de la tarjeta DE2 aplicadas a nuestro proyecto.

4.1.1 Escenario A: Envío fallido de la onda generada debido a la selección incorrecta del puerto de comunicación del computador.

Una vez que se haya montado todo el sistema, es importante hacer una correcta selección del puerto de comunicación entre el computador y la tarjeta DE 2. Esta selección se la hace desde el Generador de Funciones realizado en MATLAB.

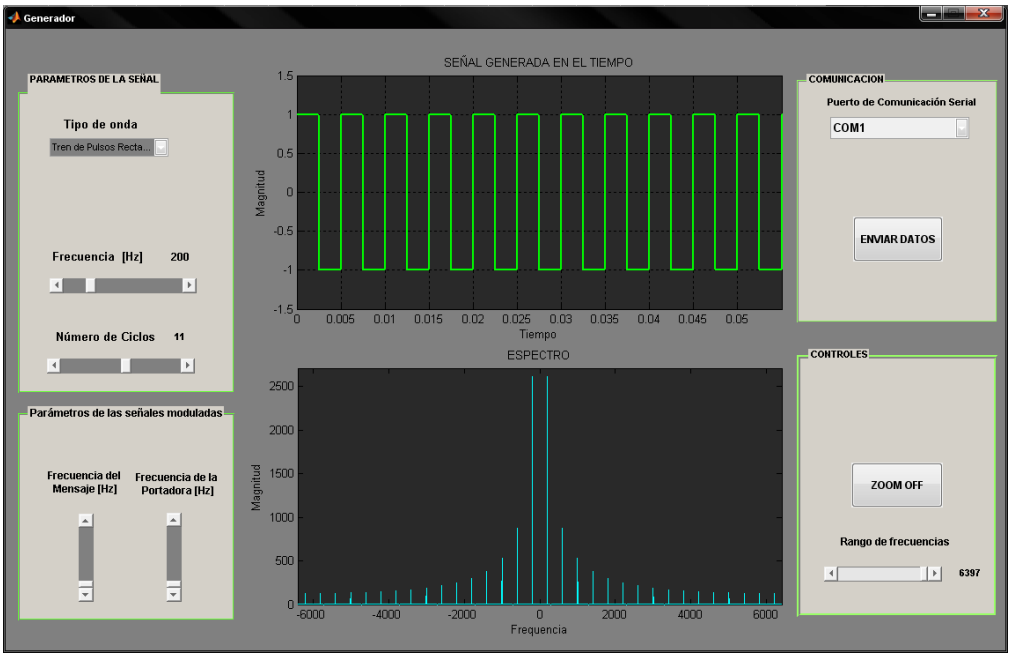


Figura 4-1 Generación de la onda “Tren de Pulsos Rectangulares”

En la figura 4-1 se ha generado la onda “Tren de Pulsos Rectangulares” de 200Hz de frecuencia, se puede ver que se ha seleccionado como puerto de comunicación COM1.

Luego de haber generado esta onda debemos presionar el botón ***“ENVIAR DATOS”*** para el envío de los datos muestreados hacia la tarjeta DE2, al presionar este botón el generador de funciones abre el puerto de comunicación COM1.

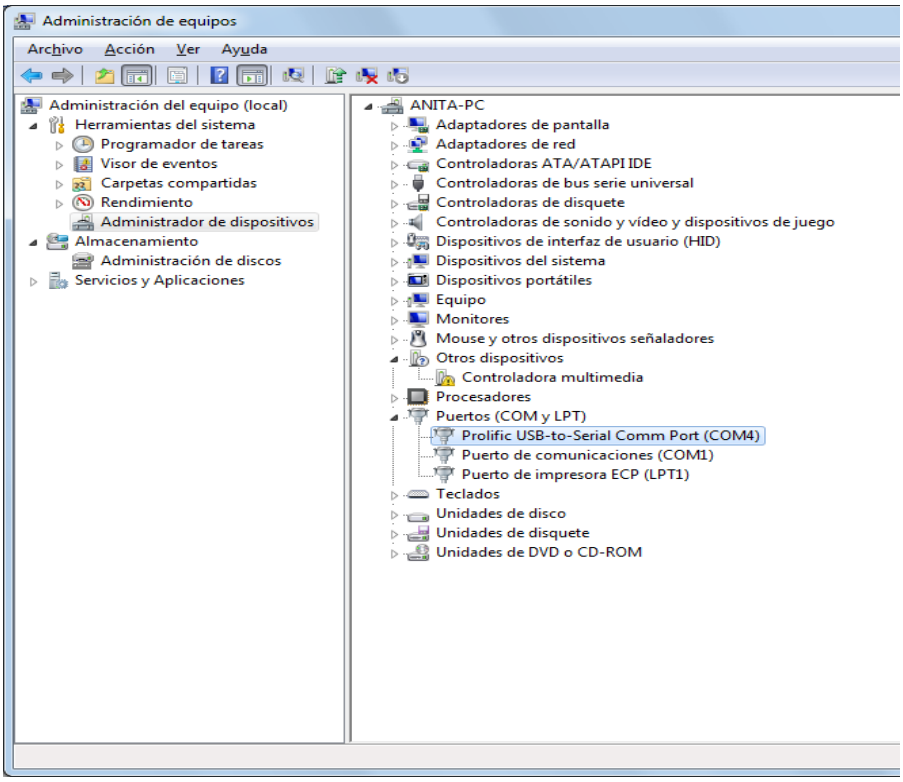


Figura 4-2 Ventana "Administración de equipos" de Windows

Suponiendo que el puerto habilitado por Windows es el puerto COM4, mostrado en la figura 4-2, en este caso el generador de funciones no podrá abrir el puerto desde MATLAB debido a que en el generador se ha seleccionado el COM1 (puerto por defecto) y en este escenario se produciría un error.

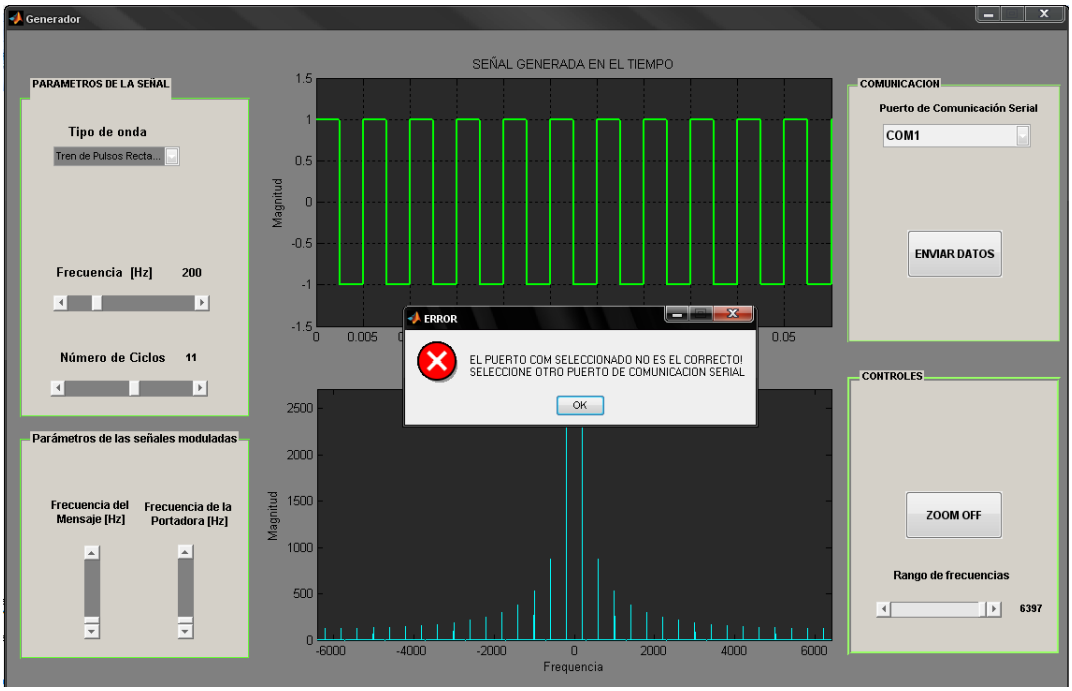


Figura 4-3 Ventana de Error del Generador de Funciones

En la figura 4-3 se observa que el generador de funciones abre una ventana de error con el mensaje:

“EL PUERTO COM SELECCIONADO NO ES EL CORRECTO! SELECCIONE OTRO PUERTO DE COMUNICACIÓN SERIAL”

Como se esperaba el generador de funciones nos mostró un fallo en la comunicación con la tarjeta DE2.

Para este escenario el monitor VGA no muestra en la pantalla ninguna señal ya que no recibió datos para procesar. La pantalla permanece en

negro debido a que no se ha realizado ninguna comunicación seril exitosa desde que se activó el sistema y por tal razón no posee datos que graficar.

4.1.2 Escenario B: Envío exitoso de la onda generada debido a la selección correcta del puerto de comunicación del computador.

En este escenario se analiza los pasos para tener una comunicación exitosa entre el computador y la tarjeta DE2.

Primero generamos la onda que se desee enviar a la tarjeta DE2 como se muestra en la figura 4-1.

El siguiente paso a seguir es seleccionar correctamente el puerto de comunicación serial usando el elemento de interfaz gráfica ***"Puerto de Comunicación Serial"*** de nuestro generador de funciones.

Basándonos en lo propuesto en el escenario A el puerto correcto habilitado es COM4, por lo que debemos seleccionar este puerto tal y como se muestra en la figura 4-4.

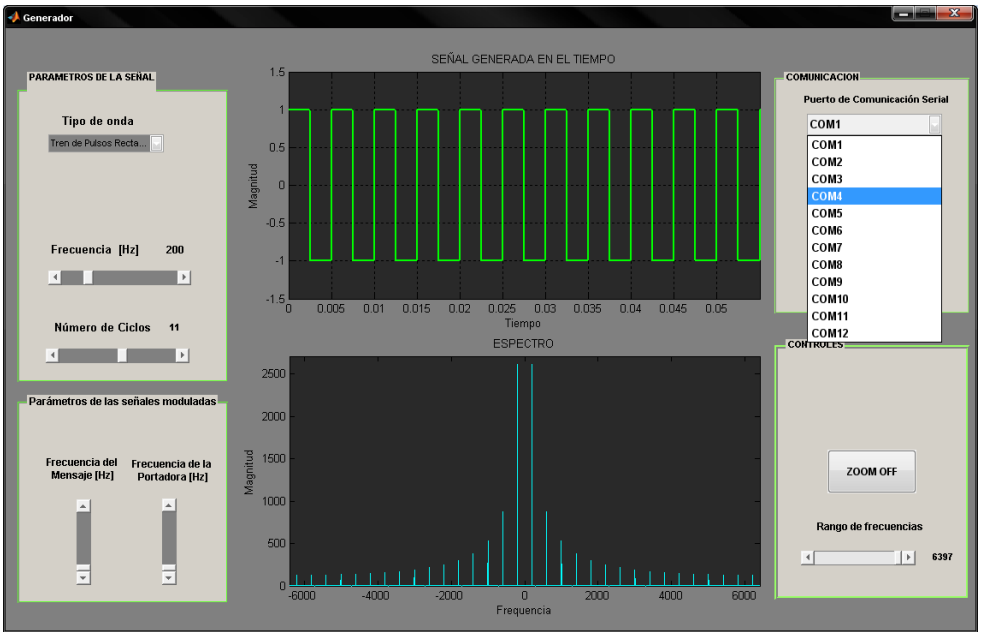


Figura 4-4 Selección de puerto "COM4"

El siguiente paso a seguir es presionar el botón ***"ENVIAR DATOS"*** de nuestro generador de funciones.

Esta vez tendremos una comunicación exitosa ya que se ha corregido el error del escenario A, después de enviar todos los 1024 datos de muestra, la tarjeta estará lista a procesar los datos que ha recibido mostrando los resultados observados en la figura 4-5.

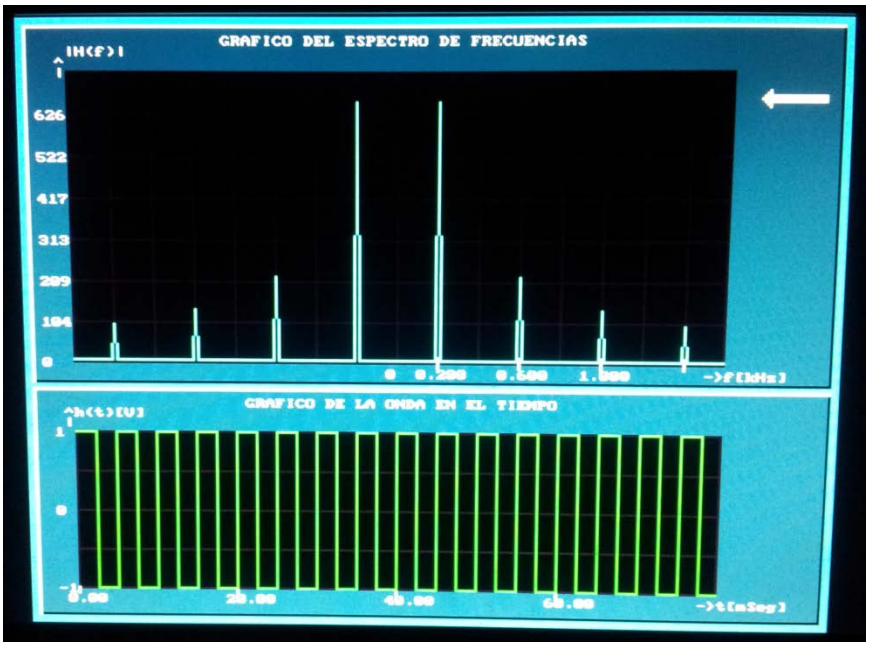


Figura 4-5 Señales graficadas en la Pantalla del monitor VGA

En la figura 4-5 vemos que la gráfica de color verde corresponde a una señal “Tren de Pulsos Rectangulares” y la de color azul turquesa es su respectivo espectro de frecuencias que ha sido resultado de haber usado el algoritmo de la Transformada Rápida de Fourier (FFT).

Comparando las figuras 4-4 y 4-5 podemos darnos cuenta que se ha hecho un correcto proceso de los datos en la tarjeta DE2, ya que las ondas graficadas en la pantalla del monitor VGA coinciden con las gráficas enviadas por nuestro generador de funciones, es decir, se obtuvo el resultado esperado previo al envió de los datos hacia la tarjeta.

4.1.3 Escenario C: Envío fallido de una nueva onda generada debido a la selección incorrecta del puerto de comunicación del computador si previamente se obtuvo éxito en lo planteado en el escenario B.

En este escenario se plantea la siguiente pregunta: ¿Qué ocurrirá si se produce una nueva comunicación serial fallida?

Esta nueva comunicación fallida puede darse debido a que se ha seleccionado nuevamente un puerto de comunicación incorrecto, sea esto producto de un acto premeditado o simplemente porque se cerró accidentalmente la aplicación en MATLAB que nos genera la onda y no se cambió el puerto de comunicación por defecto al puerto de comunicación correcto (*COM4* según lo planteado en los escenarios A y B) de nuestro generador al ejecutarlo nuevamente.

Suponiendo que se cerró accidentalmente nuestra aplicación que genera la señal muestreada, inmediatamente procedemos a abrirla y luego configuramos los parámetros de señal observado en la figura 4-6.

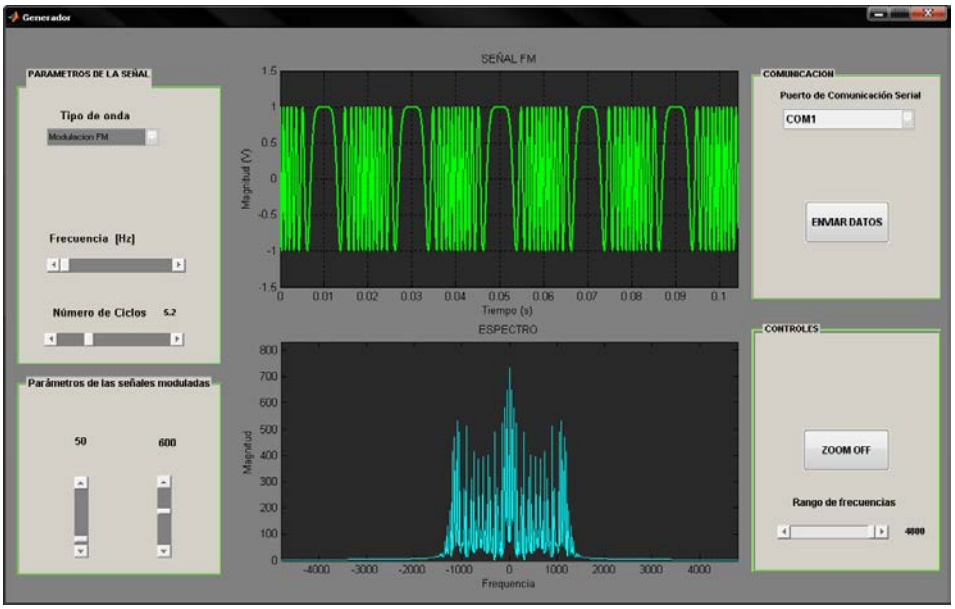


Figura 4-6 Generación de la onda "FM" sin cambiar el puerto de comunicación configurado por defecto

En la figura 4-6 se puede apreciar que accidentalmente se ha dejado configurado el *COM1* como puerto de comunicación serial lo que produciría un error en la comunicación serial suponiendo que se ha habilitado el *COM4* para la correcta comunicación por parte del sistema operativo Windows, tal y como se ha planteado en los escenario A y B.

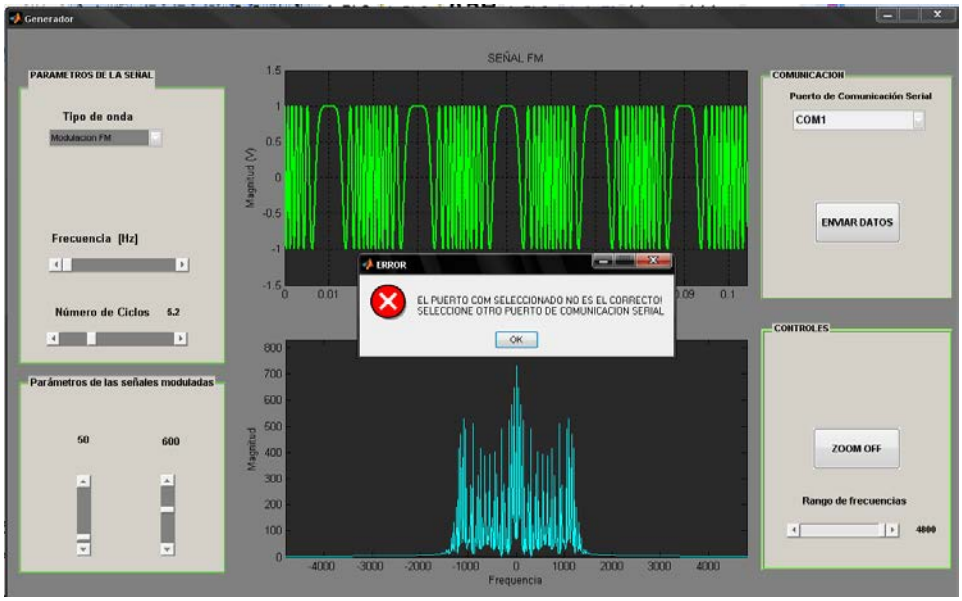


Figura 4-7 Ventana de Error del Generador de Funciones

Como se esperaba en la figura 4-7 se observa que el generador de funciones muestra un mensaje de error en la comunicación, como ya se analizó en el escenario A, pero la diferencia es que ahora en la pantalla del monitor VGA se muestra lo que había graficado previamente en el desarrollo del escenario B (comunicación serial exitosa), como se observa en la figura 4-8.

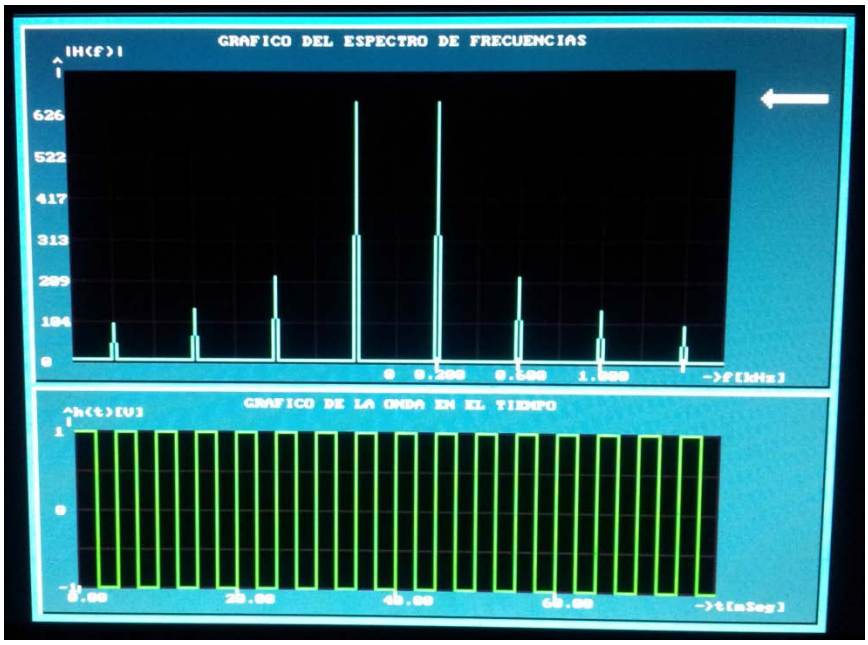


Figura 4-8 Señales graficadas en la Pantalla del monitor VGA

En la figura 4-8 vemos que la pantalla no ha cambiado, es decir, muestra la señal que se envió la última vez que se obtuvo una transmisión exitosa (Escenario B).

Para corregir este error debemos seguir los pasos planteados en el escenario B, lo que tendremos como resultado será lo observado en la figura 4-9 mostrada a continuación.

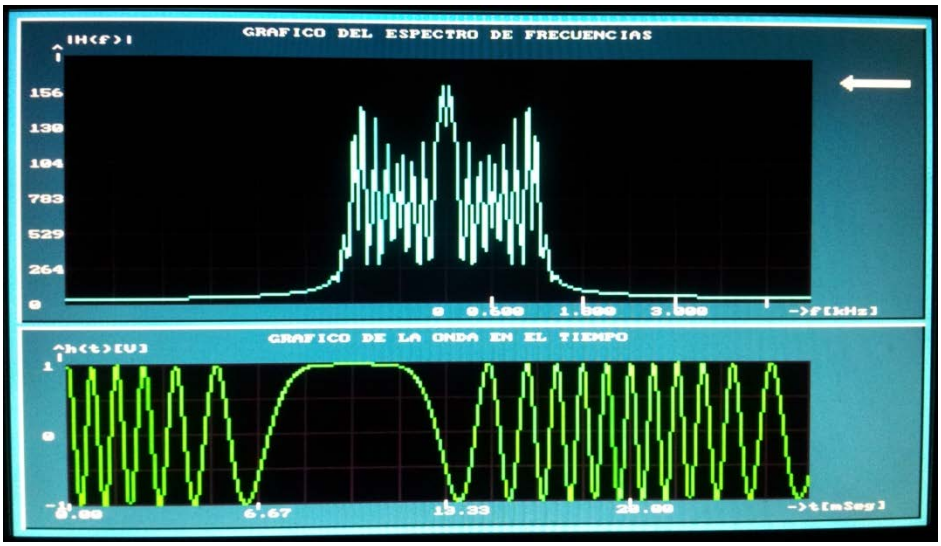


Figura 4-9 Señales graficadas en la Pantalla del monitor VGA luego de haber corregido el error de comunicación serial

Vemos en la figura 4-9 que estos resultados si coinciden con los esperados de acuerdo a lo graficado en nuestro generador de funciones (ver figura 4-8), esto es una señal FM con frecuencia de mensaje de 50 Hz y frecuencia de portadora de 600 Hz.

4.1.4 Escenario D: Envío exitoso de una onda generada y comparación numérica de los resultados mostrados en la pantalla del monitor VGA.

En este escenario vamos a hacer una comparación numérica de los resultados mostrados en la pantalla del monitor VGA con lo mostrado en nuestra aplicación generadora de funciones desarrollada en MATLAB.

Primero generamos y enviamos exitosamente la onda que se observa en la figura 4-10.

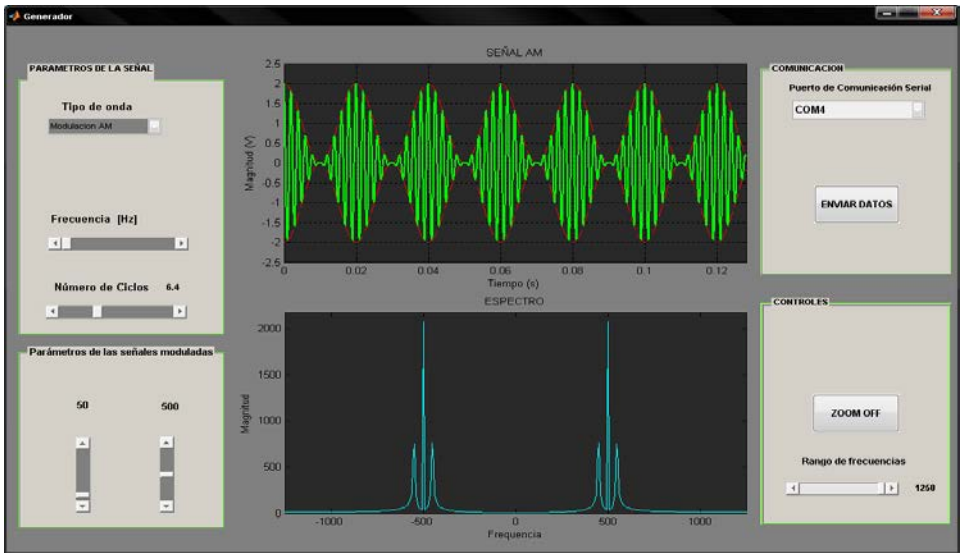


Figura 4-10 Generación de la onda "AM"

En la figura 4-10 se ha generado y enviado una señal AM 50 Hz de frecuencia de mensaje y 500 Hz de frecuencia de portadora.

La figura 4-11 muestra que la señal generada tiene un valor de magnitud de 2000 unidades para cuando la frecuencia es 500 Hz y -500Hz, mientras que en las bandas laterales se muestra una amplitud de aproximadamente 700 unidades para cuando la frecuencia es igual a 550Hz, en teoría estos valores de amplitud deberían ser números infinitos o números muy grandes, pero en la práctica el generador no grafica estos valores teóricos debido a que la onda en el dominio del tiempo esta muestreada a una frecuencia de muestreo de:

$$fs = 64 * fp$$

Donde *fs* es la frecuencia de muestreo de la onda *AM* generada y *fp* es la frecuencia de la portadora de esta señal.

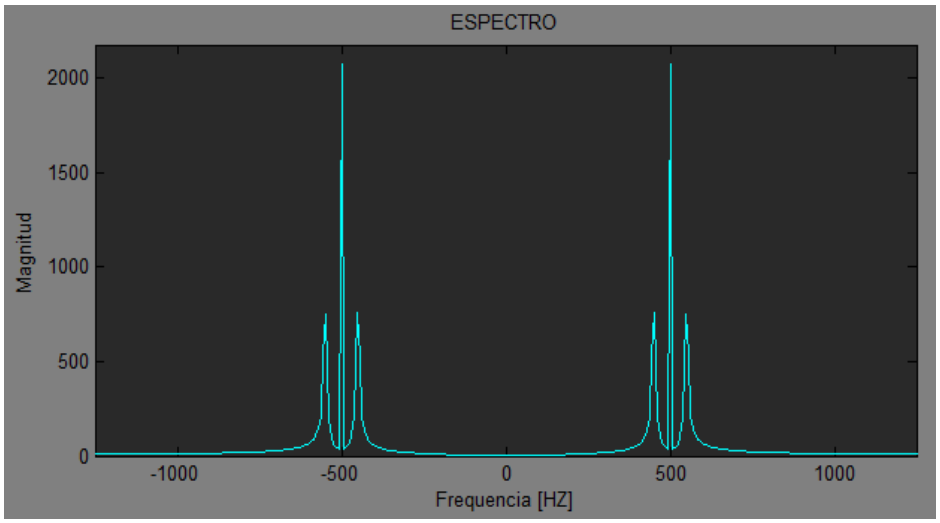


Figura 4-11 Espectro de frecuencias de la señal generada

Si aumentásemos la relación entre *fs* y *fp* de 64 veces a un número mayor, entonces obtendríamos una señal *AM* más pura debido a un aumento en la frecuencia de muestreo, lo que haría que los valores de impulsos mostrados en la figura 4-11 se acerquen más a los valores teóricos. Hacer esto no es muy práctico, ya que al aumentar la frecuencia de muestreo también aumentaría la cantidad de datos muestreados, es decir, tendríamos más de 1024 datos que transmitir lo que haría lenta la transmisión total de los datos quitándole así la versatilidad a nuestro proyecto en cuanto a lo que a velocidad de transmisión respecta.

El resultado obtenido es el observado en la figura 4-12.

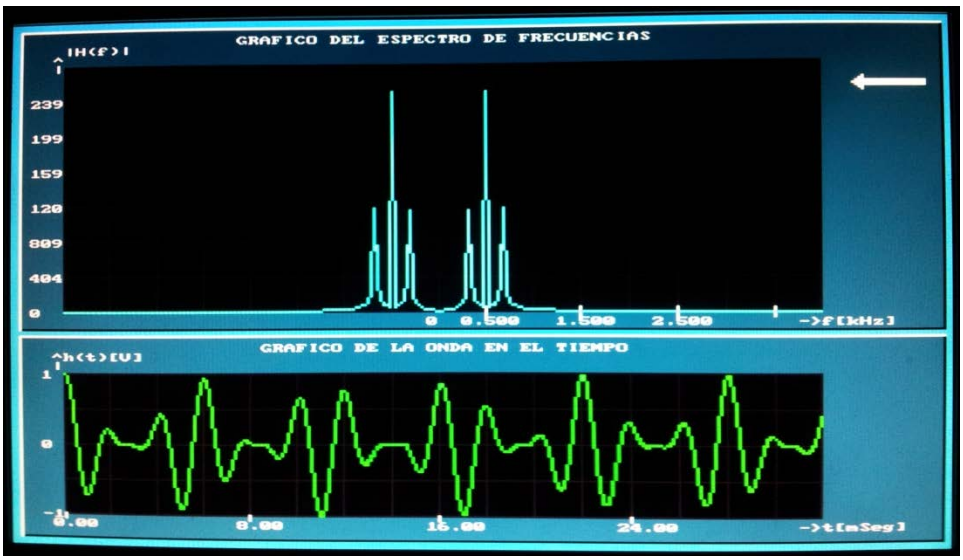


Figura 4-12 Señales graficadas en el monitor VGA

En la figura 4-12 vemos que la gráfica dibujada de color verde corresponde a una señal *AM* y la de color azul turquesa es su respectivo espectro de frecuencias que ha sido resultado de haber usado el algoritmo de la Transformada Rápida de Fourier (FFT).

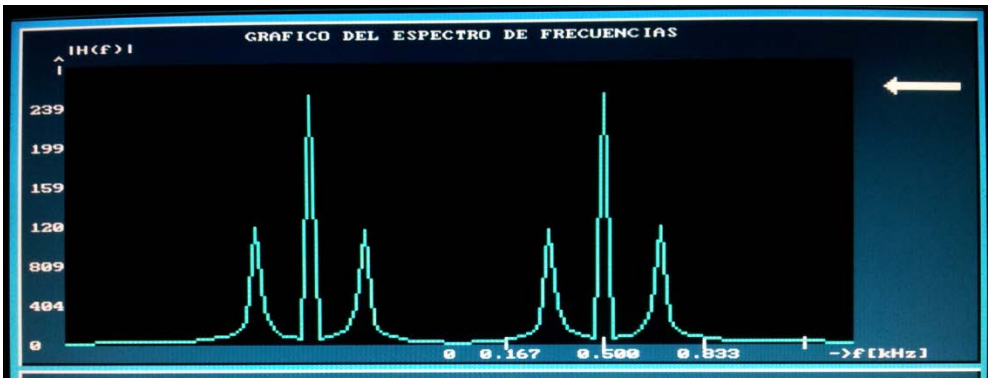


Figura 4-13 Espectro de frecuencias graficado en la Pantalla del monitor VGA

En la figura 4-13 vemos nuevamente el espectro de la señal pero con zoom aumentado para poder apreciar mejor los valores de frecuencia (para una explicación mas detallada del funcionamiento del zoom en la tarjeta referirse al anexo C – página 140). Podemos observar de la figura 4-13 que la amplitud para frecuencias de 500Hz y -500Hz son valores de aproximadamente 250 unidades, cuando por lo menos deberían ser de 2000 unidades como se plantea en la figura 4-11. Esta notoria diferencia es porque el proyecto realiza la transformada Rápida de Fourier de las 1024 muestras recibidas serialmente obteniendo así todos los puntos de frecuencias que teóricamente debería tener, pero

con la limitante de que no se dispone de tantos pixeles de resolución VGA para graficar estos 1024 puntos obtenidos al aplicar el algoritmo FFT, procede a hacer una compresión de la señal en el dominio de la frecuencia (eje horizontal del espectro), lo que implica la perdida de ciertos datos, es decir, se pasa de tener 1024 datos a tan solo 256 datos a ser graficados en la pantalla del monitor VGA de una resolución de 320x240 pixeles.

4.1.5 Escenario E: Pruebas de rendimiento del procesador de la Tarjeta DE2 aplicadas a nuestro proyecto.

En este escenario se toman datos del tiempo que demora la tarjeta DE2 en realizar un fragmento de código de nuestro proyecto. Para esto se hace uso del Interval Timer Core de Altera, configurado como contador descendente de 32 bits, desde SOPC BUILDER, que trabaja con la frecuencia del sistema, con el fin de tener datos de tiempo confiables y de gran precisión.

Se obtendrán resultados de tiempo de procesamiento para las distintas señales del generador de MATLAB, tanto en la decodificación de los datos recibidos como en el cálculo de la FFT.

```
*(TimerTimeoutL)=0xffff;
*(TimerTimeoutH)=0xffff;
*(TimerControl)=4;

decodificarDatos(real, sig, rsig);

*(TimerControl)=8;
*(TimerSnapshotL)=0;
*(TimerSnapshotH)=0;
numclow = 0xffff & *(TimerSnapshotL);
numchigh = 0xffff & *(TimerSnapshotH );
numclks = 0xffffffff & (numclow + (numchigh << 16));
printf("Tiempo de procesamiento de la decodificacion de los datos
recibidos: %f segundos\n", (float)((4294967295 - (float)(numclks)) / 50000000));
```

Figura 4-14 Fragmento de código 1 para pruebas de cálculo de capacidad de procesamiento

En la figura 4-14 se ha implementado las líneas de código para poner en marcha el conteo del Timer. El código muestra la función *decodificarDatos*. De esta manera mediremos el tiempo que le toma al procesador realizar la decodificación de los datos recibidos en código PCM.

```
*(TimerTimeoutL)=0xffff;
*(TimerTimeoutH)=0xffff;
*(TimerControl)=4;

//funcion para calcular la FFT
fft2 (n, exp, real, imag);

*(TimerControl)=8;
*(TimerSnapshotL)=0;
*(TimerSnapshotH)=0;
numclow = 0xffff & *(TimerSnapshotL);
numchigh = 0xffff & *(TimerSnapshotH );
numclks = 0xffffffff & (numclow + (numchigh << 16));
printf("Tiempo de procesamiento de la FFT de los datos:
%f segundos\n", (float)((4294967295 - (float)(numclks)) / 50000000));
```

Figura 4-15 Fragmento de código 2 para pruebas de cálculo de capacidad de procesamiento

La siguiente función seleccionada para medir la capacidad del procesador es *fft2*, la misma que se encarga de calcular la transformada rápida de Fourier de los datos decodificados. La figura 4-15 muestra el fragmento de código para llevar a cabo esta tarea.

Desde el Generador de Funciones de MATLAB se envian diferentes tipos de señales, para así realizar una comparación de los resultados obtenidos. Como se muestra en la tabla 4-1 en la primera prueba enviamos una señal sinusoidal de frecuencia 100 Hz., para una segunda prueba la señal enviada es una Diente de Sierra con frecuencia de 150 Hz., y como prueba final generamos una señal modulada en frecuencia, con portadora de 1000 Hz y frecuencia de mensaje de 100 Hz.

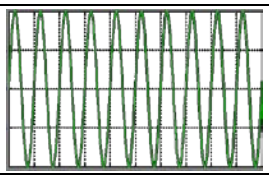
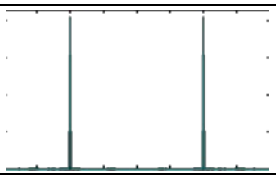
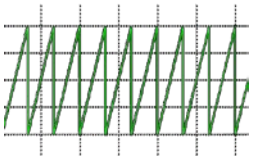
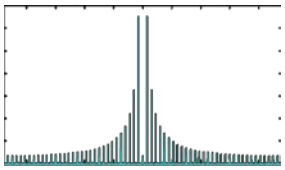
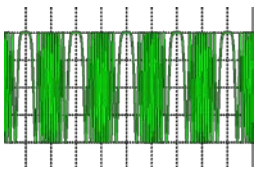
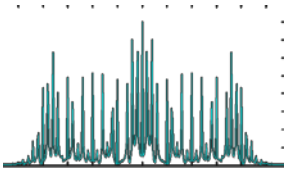
Señal	Frecuencia (Hz)	Gráfico	Espectro	Tiempo (s)
Seno	100			Decod 0.258555 FFT2 1.668567
Diente de Sierra	150			Decod 0.258621 FFT 1.668787
FM	f_c 1000 f_m 100			Decod 0.258796 FFT 2.309258

Tabla 4-1 Resultados del tiempo de procesamiento

Para cada una de las señales transmitidas a la tarjeta los resultados arrojados por el Timer se muestran en la tabla 4-1 en la columna Tiempo (s), se puede apreciar que el tiempo de procesamiento para una señal sinusoidal de la decodificación de datos recibidos es de 0.258555 segundos y el del cálculo de la FFT es de 1.6668567 segundos. El resultado se debe a la cantidad de datos que se necesitan decodificar y además las transformaciones de tipo de dato necesarias para realizar un correcto cálculo de la transformada de Fourier.

En la tabla 4-2 se muestran los resultados resumidos de los tiempos de procesamiento obtenidos en las pruebas con la función *decodificarDatos*.

Señal	# Líneas de Código	# Lazos	# Iteraciones del lazo	Tiempo (s)
Seno	9	2	1024	0.258555
Diente de sierra	9	2	1024	0.258621
FM	9	2	1024	0.258796

Tabla 4-2 Tiempos procesamiento función decodificarDatos

Podemos observar como para las tres señales los tiempos son muy similares, ya que esta función solo se encarga de decodificar los datos recibidos y dejarlos listos para procesamientos posteriores. Cabe recalcar que para las distintas señales enviadas, esta función emplea la misma cantidad de líneas de código, lazos e iteraciones de lazos.

En la tabla 4-3 se muestran los resultados resumidos de los tiempos de procesamiento obtenidos en las pruebas con la función *FFT2*.

Señal	# Líneas de Código	# Lazos	# Iteraciones del lazo	Tiempo (s)
Seno	36	5	1024/variable	1.666.856
Diente de sierra	36	5	1024/variable	1.668.787
FM	36	5	1024/variable	2.309.258

Tabla 4-3 Tiempos de procesamiento para la función FFT2

Podemos observar como para las dos primeras señales los tiempos son similares, exceptuando el caso de la señal modulada en frecuencia, la cual demora más. Este resultado es de esperarse debido a que el espectro de una señal FM contiene una mayor cantidad de bandas en comparación a los espectros de las señales seno y diente de sierra, lo que provoca que la tarjeta utilice una mayor cantidad de recursos para procesar estos datos. Este cambio en el tiempo también se debe a los lazos de la función que son de duración variable dependiendo de los datos a procesar.

CONCLUSIONES

1. Se pudo desarrollar un código en NIOS II que facilitó el procesamiento y visualización de una señal en un monitor VGA, usando el algoritmo de la Transformada Rápida de Fourier (FFT) para la comparación de resultados en las distintas etapas del proyecto.

2. Para poder demostrar el funcionamiento de este proyecto fue necesario desarrollar en MATLAB una aplicación para la generación y envío serial de una señal muestreada hacia la tarjeta DE2, debido a los costos que implicaría el usar un convertidor Analógico – Digital (ADC) para adquirir físicamente una señal continua de tiempo. El previo uso de esta aplicación fue de mucha ayuda, ya que nos proporcionó los datos que se debería obtener al procesar y graficar en un monitor VGA los datos de la señal generada, lo que facilitó las pruebas y el análisis de los resultados obtenidos.

3. El uso de comunicación serial para transmitir los datos generados desde el computador a la tarjeta DE2 supuso un incremento en el tiempo de procesamiento y limitó la cantidad de datos que se utilizaron para calcular la transformada de Fourier, dando como resultado una diferencia

entre los espectros obtenidos en MATLAB con los graficados por la tarjeta en el monitor VGA.

4. La velocidad de procesamiento del algoritmo para el cálculo de la FFT es relativamente lento, llegando a demorar cerca de un minuto y medio en calcular la transformada de Fourier para señales modulas. Sin embargo, para fines educativos es aceptable.

5. El desarrollo de nuestro proyecto en lenguaje C para NIOS II trajo limitaciones principalmente en la velocidad de procesamiento, ya que de haber utilizado más descripción de hardware, estos tiempos se hubiesen reducido considerablemente. Las soluciones en hardware resultan siempre más eficientes que las basadas en software.

RECOMENDACIONES

1. Se recomienda configurar de manera correcta los diferentes parámetros de señal en el software generador de funciones desarrollado en MATLAB, sobre todo el parámetro *"Puerto de Comunicación Serial"* ubicado en el panel *"COMUNICACIÓN"* de nuestro generador de funciones.

2. Como futura mejora, se recomienda el uso de un convertidor analógico – digital, el cual proporcionaría una adquisición física y en tiempo real de una onda adquirida de un generador de funciones o un acoplador para una antena que receptaría una señal del ambiente. Lo que implicaría también un incremento en la velocidad con la que la tarjeta DE2 adquiere los datos a procesar.

3. El uso de un puerto paralelo mejoraría la velocidad de la transmisión de datos en lugar de comunicación serial, ya que esta envía una mayor cantidad de datos por unidad de tiempo.

4. Para una mayor velocidad de procesamiento de los datos, se recomienda usar un Core desarrollado en lenguaje de descripción de hardware, ya

que este tipo de soluciones proporcionan mejores tiempos de respuesta comparados a los tiempos obtenidos en funciones netamente de software, desarrolladas para ejecutarse en el procesador NIOS II.

5. La resolución del monitor VGA se puede mejorar si se utiliza un Core más poderoso, que configure cantidad de pixeles mayores a las que se usan en nuestro proyecto, ya que la tarjeta DE2 soporta una resolución 1600x1200 mientras que con el Core VGA usado solo se llega a 640 x 320.

6. El uso de un Core PLL proporcionaría una mayor velocidad de procesamiento al aumentar la frecuencia del reloj del sistema, teniendo en cuenta que como valor máximo se tiene 200 MHz para esta tarjeta específica.

BIBLIOGRAFIA

[1] Loomis John, Altera's Development and Education (DE2) Board, http://www.johnloomis.org/altera/, fecha de consulta junio 2012

[2] ALTERA, DE2 Development and Education Board – Getting started Guide, http://users.ece.gatech.edu/~hamblen/DE2/DE2%20Reference%20Manual.pdf, fecha de consulta junio 2012, capítulo 2 – páginas 5 y 6

[3] ALTERA, Cyclone II FPGA, http://www.digchip.com/datasheets/parts/datasheet/033/EP2C35F672C6.php, fecha de consulta junio 2012

[4] ALTERA, JTAG Debug Module, http://www.altera.com/devices/processor/nios2/benefits/ni2-jtag-debug.html, fecha de consulta junio 2012

[5] Radhakrishnan Sudhakar, Effective Video Coding For Multimedia Applications, InTech, fecha de consulta junio 2012, capítulo 12 – página 236

[6] The University of Adelaide, Jean Baptiste Joseph Fourier, http://ebooks.adelaide.edu.au/f/fourier/joseph/index.html, fecha de consulta julio 2012

[7] CMLAB, Fast Fourier Transform (FFT), http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html, fecha de consulta julio 2012

[8] MathWorks, Matlab – The language of Technical Computing, http://www.mathworks.com/products/matlab/, fecha de consulta julio 2012

[9] Pong P. Chu, Embedded SOPC design with NIOS II Processor and VHDL examples, WILEY Editorial, 2011, fecha de consulta junio 2012, capítulo 3 – página 29

[10] Pong P. Chu, Embedded SOPC design with NIOS II Processor and VHDL examples, WILEY Editorial, 2011, fecha de consulta junio 2012, capítulo 2 – página 11

[11] ALTERA, DE2 Development and Education Board, http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html, fecha de consulta junio 2012

[12] ALTERA, DE2 Development and Education Board – User Manual, ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf, fecha de consulta junio 2012, capítulo 1 – página 2

[13] ALTERA, DE2 Development and Education Board - Getting started Guide, http://users.ece.gatech.edu/~hamblen/DE2/DE2%20Reference%20Manual.pdf, fecha de consulta junio 2012, capítulo 2 – página 4

[14] Pong P. Chu, Embedded SOPC design with NIOS II Processor and VHDL examples, WILEY Editorial, 2011, fecha de consulta junio 2012, capítulo 3 – páginas 21 y 22

[15] Pong P. Chu, Embedded SOPC design with NIOS II Processor and VHDL examples, WILEY Editorial, 2011, fecha de consulta junio 2012, capítulo 8 – página 181

[16] ALTERA, Altera University Program – Learning Through Innovation, http://www.altera.com/education/univ/unv-index.html, fecha de consulta agosto 2012

ANEXOS

ANEXO A

CODIGO FUENTE EN LENGUAJE C

```
/*
 * pr.c
 *
 *  Creado: 02/01/2013
 *  Autores: CRISTÓBAL ANDRÉS CANDEL LAYANA
 *               JOSÉ ISRAEL MAYORGA BAYAS
 */

#include “stdio.h”
#include “stdlib.h”
#include “math.h”
#include “system.h”
#include “altera_up_avalon_video_character_buffer_with_dma.h”
#include "altera_up_avalon_video_character_buffer_with_dma_regs.h"
#include "altera_up_avalon_video_pixel_buffer_dma.h"
#include "altera_up_avalon_rs232.h"

alt_up_char_buffer_dev *cvga;
alt_up_pixel_buffer_dma_dev *pvga;
alt_up_rs232_dev *uart;

#define rs232bytes_available(c) (c>>16)
#define pRS232 ((volatile int*) 0x10001010)
#define RS232_UART_CONTROL ((volatile int*) (0x10001010+4))
#define key ((volatile int*) (0x10000050))
#define sw  ((volatile int*) (0x10000040))
#define ledR ((volatile int*) (0x10000000))
#define ledG ((volatile int*) (0x10000010))
//TIMER
#define Timer 0x10002000
//
#define TimerStatus ((volatile short*) (Timer))
#define TimerControl ((volatile short*) (Timer+4))
#define TimerTimeoutL ((volatile short*) (Timer+8))
#define TimerTimeoutH ((volatile short*) (Timer+12))
#define TimerSnapshotL ((volatile short*) (Timer+16))
#define TimerSnapshotH ((volatile short*) (Timer+20))



void write_pixel (int, int, short);
void VGA_text (int, int, char *);
void VGA_box (int, int, int, int, short);
void fft (int n, int m, double x[], double y[]);
void magnitud (int n, double x[], double y[], double r[]);
void entero (int n, double m[], int r[]);
int position (int n, int m[]);
int maximo (int n, int m[]);
```

```
int getcharRS232();
void getDatos (double real[]);
void decodificarDatos (double real[], double sig[], int rsig[]);
void dibujarespectro (int fme, int max, double df, int zmfft, int d, int mag[]);
void graficodeTiempo (int fme, int zm, int r[]);
void fft2 (int n, int m, double x[], double y[]);


unsigned int rs232word;


int main (void)
{
		int i, j, t, fme, zm, d, zmfft;
		int xp=29;
		double real[1024], real2[1024], imag[1024], sig[1024];
		int rmag[1024], rmag2[1024], rsig[1024];
		int n, exp, max, pm, esc, v, yp, pi, pf, temp;
		double df, f, t1;
		//TIMER
		unsigned long numclks,numchigh,numclow;//Timer
		//


		uart = alt_up_rs232_open_dev("/dev/Serial_Port");
		cvga = alt_up_char_buffer_open_dev("/dev/VGA_Char_Buffer");
		pvga = alt_up_pixel_buffer_dma_open_dev("/dev/VGA_Pixel_Buffer");

		alt_up_char_buffer_clear(cvga);
		alt_up_pixel_buffer_dma_clear_screen(pvga, 0);
		alt_up_pixel_buffer_dma_clear_screen(pvga, 1);

		alt_up_rs232_enable_read_interrupt(uart);

		alt_up_pixel_buffer_dma_draw_box(pvga, 1, 1, 319, 249, 0x1DDF, 0); //Color de Fondo
		alt_up_pixel_buffer_dma_draw_box(pvga, 4, 4, 316, 141, 0x1AAF, 0); //Color de Fondo
		alt_up_pixel_buffer_dma_draw_hline(pvga, 5, 316, 141, 0x0000, 0);
		alt_up_pixel_buffer_dma_draw_vline(pvga, 316, 5, 141, 0x0000, 0);
		alt_up_pixel_buffer_dma_draw_hline(pvga, 5, 316, 4, 0xFFFF, 0);
		alt_up_pixel_buffer_dma_draw_vline(pvga, 4, 4, 141, 0xFFFF, 0);

		alt_up_pixel_buffer_dma_draw_box(pvga, 4, 144, 316, 236, 0x1AAF, 0); //Color de Fondo
		alt_up_pixel_buffer_dma_draw_hline(pvga, 5, 316, 236, 0x0000, 0);
		alt_up_pixel_buffer_dma_draw_vline(pvga, 316, 144, 236, 0x0000, 0);
		alt_up_pixel_buffer_dma_draw_hline(pvga, 5, 316, 144, 0xFFFF, 0);
		alt_up_pixel_buffer_dma_draw_vline(pvga, 4, 144, 236, 0xFFFF, 0);

		alt_up_pixel_buffer_dma_draw_box(pvga, 20, 21, 275, 133, 0x0000, 0); //Borra grafico
		alt_up_pixel_buffer_dma_draw_box(pvga, 20, 161, 275, 225, 0x0000, 0);

		//Dibuja Cuadricula para Gráfica en la Frecuencia
		alt_up_char_buffer_string (cvga, "GRAFICO DEL ESPECTRO DE FRECUENCIAS", 20, 2);
		alt_up_char_buffer_string (cvga, "|H(f)|", 5, 3);
		alt_up_char_buffer_string (cvga, "^", 4, 4);
		alt_up_char_buffer_string (cvga, "|", 4, 5);
		alt_up_char_buffer_string (cvga, "->f[Hz]", 67, 34);
		alt_up_pixel_buffer_dma_draw_rectangle (pvga, 20, 21, 275, 133, 0x1863, 0);

		for (i = 1; i <= 6; i++)
			alt_up_pixel_buffer_dma_draw_hline(pvga, 20, 275, 21 + (i*16), 0x1863, 0);

		for (i = 1; i <= 15; i++)
```

```
		alt_up_pixel_buffer_dma_draw_vline(pvga, 20 + (i*16), 21, 133, 0x1863, 0);

	//Dibuja Cuadricula para Gráfica en el Tiempo
	alt_up_char_buffer_string (cvga, "GRAFICO DE LA ONDA EN EL TIEMPO", 22, 37);
	alt_up_char_buffer_string (cvga, "h(t)[V]", 5, 38);
	alt_up_char_buffer_string (cvga, "^", 4, 38);
	alt_up_char_buffer_string (cvga, "|", 4, 39);
	alt_up_char_buffer_string (cvga, "->t[mSeg]", 67 , 57);
	alt_up_pixel_buffer_dma_draw_rectangle(pvga, 20, 161, 275, 225, 0x1863, 0);

	for (i = 1; i <= 3; i++)
		alt_up_pixel_buffer_dma_draw_hline(pvga, 20, 275, 161 + (i*16), 0x1863, 0);

	for (i = 1; i <= 15; i++)
		alt_up_pixel_buffer_dma_draw_vline(pvga, 20 + (i*16), 161, 225, 0x1863, 0);
	If (*sw == 1)
		*ledR = 1;
	else
		*ledR = 0;

	while (1)
	{

		//encerar matriz imaginaria para evitar errores
		for (i = 0; i < 1024; i++)
			imag[i] = 0;

		if (*sw == 1)
			*ledR = 1;
		else
			*ledR = 0;

		printf ("Esperando Datos:\n");

		i = 0;

		while (1)
		{
			If (*sw == 1)
			{
				*ledR = 1;
				alt_up_pixel_buffer_dma_draw_box(pvga, 288, 30, 310, 32, 0xFFFF, 0);
				alt_up_pixel_buffer_dma_draw_line(pvga, 286, 31, 289, 28, 0xFFFF, 0);
				alt_up_pixel_buffer_dma_draw_line(pvga, 287, 31, 289, 29, 0xFFFF, 0);
				alt_up_pixel_buffer_dma_draw_line(pvga, 286, 31, 289, 34, 0xFFFF, 0);
				alt_up_pixel_buffer_dma_draw_line(pvga, 287, 31, 289, 31, 0xFFFF, 0);
				alt_up_pixel_buffer_dma_draw_line(pvga, 287, 31, 289, 33, 0xFFFF, 0);

				alt_up_pixel_buffer_dma_draw_box(pvga, 288, 140 + 30, 310, 140 + 32, 0x1AAF,
0);
				alt_up_pixel_buffer_dma_draw_line(pvga, 286, 140 + 31, 289, 140 + 28, 0x1AAF,
0);
				alt_up_pixel_buffer_dma_draw_line(pvga, 287, 140 + 31, 289, 140 + 29, 0x1AAF,
0);
				alt_up_pixel_buffer_dma_draw_line(pvga, 286, 140 + 31, 289, 140 + 34, 0x1AAF,
0);
				alt_up_pixel_buffer_dma_draw_line(pvga, 287, 31 + 140, 289, 140 + 31, 0x1AAF,
0);
				alt_up_pixel_buffer_dma_draw_line(pvga, 287, 140 + 31, 289, 140 + 33, 0x1AAF,
0);
			}
			else
```

```
{
		alt_up_pixel_buffer_dma_draw_box(pvga, 288, 30, 310, 32, 0x1AAF, 0);
		alt_up_pixel_buffer_dma_draw_line(pvga, 286, 31, 289, 28, 0x1AAF, 0);
		alt_up_pixel_buffer_dma_draw_line(pvga, 287, 31, 289, 29, 0x1AAF, 0);
		alt_up_pixel_buffer_dma_draw_line(pvga, 286, 31, 289, 34, 0x1AAF, 0);
		alt_up_pixel_buffer_dma_draw_line(pvga, 287, 31, 289, 31, 0x1AAF, 0);
		alt_up_pixel_buffer_dma_draw_line(pvga, 287, 31, 289, 33, 0x1AAF, 0);

		alt_up_pixel_buffer_dma_draw_box(pvga, 288, 140 + 30, 310, 140 + 32, 0xFFFF, 0);
		alt_up_pixel_buffer_dma_draw_line(pvga, 286, 140 + 31, 289, 140 + 28, 0xFFFF, 0);
		alt_up_pixel_buffer_dma_draw_line(pvga, 287, 140 + 31, 289, 140 + 29, 0xFFFF, 0);
		alt_up_pixel_buffer_dma_draw_line(pvga, 286, 140 + 31, 289, 140 + 34, 0xFFFF, 0);
		alt_up_pixel_buffer_dma_draw_line(pvga, 287, 31 + 140, 289, 140 + 31, 0xFFFF, 0);
		alt_up_pixel_buffer_dma_draw_line(pvga, 287, 140 + 31, 289, 140 + 33, 0xFFFF, 0);



		*ledR = 0;
		alt_up_pixel_buffer_dma_draw_box(pvga, 290, 20, 310, 22, 0x1AAF, 0);}
		if (*RS232_UART_CONTROL == 8388865 || *key != 0)
			break;
}


If (*key == 0)
{
		getDatos(real);
		//recepcion de frecuencia de muestreo
		i = 0;
		while (i < 1)
		{
			v = getcharRS232();
			if (v == 160)
				t = 0;
			else
				t = (v << 3);
			v = getcharRS232();
			if (v == 160)
				fme = t | 0;
			else
				fme = t | v;
			i++;
		}

		i = 0; n = 1024;
		//TIMER
		*(TimerTimeoutL) = 0xffff;
		*(TimerTimeoutH) = 0xffff;
		*(TimerControl) = 4;

		decodificarDatos(real, sig, rsig);

		*(TimerControl) = 8;
		*(TimerSnapshotL) = 0; //write to timer to get snapshot
		*(TimerSnapshotH) = 0; //write to timer to get snapshot
		numclow = 0xffff & *(TimerSnapshotL); //get low part
		numchigh = 0xffff & *(TimerSnapshotH ); //get high part
		numclks = 0xffffffff & (numclow + (numchigh << 16)); //assemble full number
		printf("Tiempo de procesamiento de la decodificacion de los datos recibidos: %f
		segundos\n", (float)((4294967295 - (float)(numclks)) / 50000000));
```

```
		zm = 4;

		zmfft = 1;
		d = 1;
		j = 0; i = 0; xp = 29; yp = 141; esc = 1;
		n = 1024; exp = 10;
		pi = 0; pf = 129;

		df = 64 * (float)fme / (float)n ;    //diferencial de frecuencia

		//TIMER
		*(TimerTimeoutL) = 0xffff;
		*(TimerTimeoutH) = 0xffff;
		*(TimerControl) = 4;

		//funcion para calcular la FFT
		fft2 (n, exp, real, imag);

		*(TimerControl) = 8;
		*(TimerSnapshotL) = 0; //write to timer to get snapshot
		*(TimerSnapshotH) = 0; //write to timer to get snapshot
		numclow = 0xffff & *(TimerSnapshotL); //get low part
		numchigh = 0xffff & *(TimerSnapshotH ); //get high part
		numclks = 0xffffffff & (numclow + (numchigh << 16)); //assemble full number
		printf("Tiempo de procesamiento de la FFT de los datos: %f segundos\n",
		(float)((4294967295 - (float)(numclks)) / 50000000));

		//calcula la magnitud de la FFT
		magnitud (n, real, imag, real);
		entero (n, real, rmag);

		//Obtiene el maximo y su posicion para calcular frecuencia
		max = maximo (n, rmag);
		pm = position (n, rmag);
		f = pm * df;

		for (i = 0; i < 1024 ; i++)
		{
			real2[i] = (-1) * real[i] * ((float)100/((float)max));

			t1 = real2[i]*10;
			temp = (int) t % 10;
			if (temp >= 5)
				rmag2[i] = ((int) real2[i]) + 1;
			else
				rmag2[i] = (int) real2[i];
		}
}

If (*key == 4)
		*ledG = 48;
else
		if (*key == 8)
			*ledG = 192;
		else
			*ledG = 0;

		//ZOOM  IN h(t)
If (*key == 8 && zm > 1 && *sw == 0)
		zm--;

If (*key == 4 && zm < 4 && *sw == 0)
```

```
                zm++;

            //ZOOM  IN FFT
            If (*key == 8 && zmfft >= 1 && *sw == 1 && zmfft < 5)
            {
                zmfft++;
                d = 1;
            }

            If (*key == 8 && zmfft < 1 && *sw == 1)
            {
                d--;
                zmfft = 1/d;
            }

            //ZOOM  OUT FFT
            If (*key == 4 && zmfft > 1  && *sw == 1)
            {
                zmfft--;
            }

            If (*key == 4 && zmfft <= 1  && *sw == 1)
            {
                d++;
                zmfft = 1 / d;
            }

            //VGA:
            If (*key == 0)
            {
                dibujarespectro (fme, max, df, zmfft, d, rmag2);
                graficodeTiempo (fme, zm, rsig);
            }
            else
                if (*ledR == 1)
                    dibujarespectro (fme, max, df, zmfft, d, rmag2);
                else
                    if (*ledR == 0)
                        graficodeTiempo (fme, zm, rsig);

            while (1)
            {
                If (*key == 0)
                {
                    *ledG = 0;
                    break;
                }
            }

        }
        return 0;
    }


/*****************************************************************************
 * Subroutina para Recibir los Datos
*****************************************************************************/
    void getDatos(double real[])
    {
        int v, t, i = 0, n = 1024;
        while (i < n)
        {
```

```
                        v = getcharRS232();
                        if (v == 160)
                                t = 0;
                        else
                                t = (v << 7);

                        v = getcharRS232();
                        if (v == 160)
                                real[i] = t | 0;
                        else
                                real[i] = t | v;
                        i++;
                }
        }

/*****************************************************************************
 * Decodificar Datos
*****************************************************************************/
        void decodificarDatos(double real[], double sig[], int rsig[])
        {

                int n = 1024, i;
                for (i = 0; i < n; i++)
                {
                        real[i] = 2 * ((real[i]/4095) - 0.5);
                        sig[i] = real[i] * -32;
                }
                entero (1024, sig, rsig);
        }

/*****************************************************************************
 * Graficar Espectro de Frecuencias
*****************************************************************************/
        void dibujarespectro (int fme, int max, double df, int zmfft, int d, int mag[])
        {
                int i, j, k;
                float ffx1, fy1;
                char cfx1[8], cfy1[8];
                char limpia[32] = "                    \0";
                //alt_up_char_buffer_clear(cvga);
                alt_up_pixel_buffer_dma_draw_box (pvga, 20, 21, 275, 133, 0x0000, 0); //Borra grafico

                //Dibuja Cuadricula para Gráfica en la Frecuencia
                alt_up_pixel_buffer_dma_draw_rectangle(pvga, 20, 21, 275, 133, 0x1863, 0);
                for (i = 1; i <= 6; i++)
                        alt_up_pixel_buffer_dma_draw_hline(pvga, 20, 275, 21 + (i*16), 0x1863, 0);

                for (i = 1; i <= 15; i++)
                        alt_up_pixel_buffer_dma_draw_vline(pvga, 20 + (i*16), 21, 133, 0x1863, 0);

                for (i = 0; i < 4; i++)
                        alt_up_pixel_buffer_dma_draw_vline(pvga, 164 + (i*32), 131, 135, 0xFFFF, 0);

                alt_up_char_buffer_string (cvga, "GRAFICO DEL ESPECTRO DE FRECUENCIAS", 20, 2);
                alt_up_char_buffer_string (cvga, "|H(f)|", 5, 3);
                alt_up_char_buffer_string (cvga, "^", 4, 4);
                alt_up_char_buffer_string (cvga, "|", 4, 5);
                //alt_up_char_buffer_string (cvga, "->f[Hz]", 67, 34);
                alt_up_char_buffer_string (cvga, "GRAFICO DE LA ONDA EN EL TIEMPO", 22, 37);
                alt_up_char_buffer_string (cvga, "h(t)[V]", 5, 38);
                alt_up_char_buffer_string (cvga, "^", 4, 38);
                alt_up_char_buffer_string (cvga, "|", 4, 39);
```

```
alt_up_char_buffer_string (cvga, "->t[mSeg]", 67 , 57);

//Grafica Magnitud del Espectro de Frecuencias
for (i = 0; i < 7; i++)
{
	fy1 = 0.96 * i * (((float)max) / 6);
	sprintf(cfy1, "%1.0f", fy1);
	alt_up_char_buffer_string (cvga, cfy1, 2, 33 - (4*i));
}

//Dibuja la Señal en la Frecuencia
alt_up_char_buffer_string (cvga, limpia, 37, 34);
i = 148; k = 0;
for (j = 0; j < 128; j++)
{
	If (zmfft >= 1)
	{
		alt_up_pixel_buffer_dma_draw_line(pvga, i, mag[zmfft*j]+132, i-1,
mag[zmfft*j+1]+132, 0x1FFF, 0);
		i--;
		if ((j % 32) == 0 && j!=0)
		{
			If (fme < 10 )
			{
				ffx1 = ((1 * (j * df)) - (16 * (df)));
				sprintf(cfx1, "%1.3f", (ffx1*zmfft));
				alt_up_char_buffer_string (cvga, cfx1, (33 + (8*(j/32))) , 34);
				alt_up_char_buffer_string (cvga, "->f[Hz]", 67, 34);
			}
			else
			{
				ffx1 = 0.001 * ((1 * (j * df)) - (16 * (df)));
				sprintf(cfx1, "%1.3f", (ffx1*zmfft));
				alt_up_char_buffer_string (cvga, cfx1, (31 + (8*(j/32))) , 34);
				alt_up_char_buffer_string (cvga, "->f[kHz]", 67, 34);
			}
		}
	}

	If (zmfft < 1 && (i-d) > 18)
	{
		alt_up_pixel_buffer_dma_draw_line(pvga, i, mag[j]+132, i-d, mag[j+1]+132,
0x1FFF, 0);
		i = i - d;
	}

	If (zmfft < 1)
	{
		If ((j % 32) == 0 && j!=0)
		{
			If (fme < 10 )
			{
				ffx1 = ((1 * (j * df)) - (16 * (df)));
				sprintf(cfx1, "%1.3f", (ffx1 / d));
				alt_up_char_buffer_string (cvga, cfx1, (33 + (8*(j/32))) , 34);
				alt_up_char_buffer_string (cvga, "->f[Hz]", 67, 34);

			}
			else
			{
				ffx1 = 0.001 * ((1 * (j * df)) - (16 * (df)));
				if (d > 2)
```

```
                                                                sprintf(cfx1, "%1.3f", (ffx1 / d));
                                                        else
                                                                sprintf(cfx1, "%1.3f", (ffx1 / d));
                                                        alt_up_char_buffer_string (cvga, cfx1, (31 + (8*(j/32))) , 34);
                                                        alt_up_char_buffer_string (cvga, "->f[kHz]", 67, 34);
                                                }
                                        }

                                }
                        }

                        alt_up_char_buffer_string (cvga, "0", 36 , 34);

                        i = 148; k = 0;
                        for (j = 0; j < 128; j++)
                        {
                                If (zmfft >= 1)
                                {
                                        alt_up_pixel_buffer_dma_draw_line(pvga,  i,  mag[zmfft*j]+132,  i+1,
mag[zmfft*j+1]+132, 0x1FFF, 0);
                                        i++;
                                }
                                If (zmfft < 1 && (i+d) < 278)
                                {
                                        alt_up_pixel_buffer_dma_draw_line(pvga,  i,  mag[j]+132,  i+d,  mag[j+1]+132,
0x1FFF, 0);
                                        i = i + d;
                                }
                        }

                        alt_up_pixel_buffer_dma_draw_box(pvga, 16, 10, 19, 133, 0x1AAF, 0); //Color de Fondo
                        alt_up_pixel_buffer_dma_draw_box(pvga, 276, 10, 280, 133, 0x1AAF, 0); //Color de Fondo

        }


/*******************************************************************************************
 * Graficar Señal en el Tiempo
*******************************************************************************************/
        void graficodeTiempo(int fme, int zm, int r[])
        {

                int i, j, k;
                float fx1;
                char cx1[8];
                char limpia[66] = "                                                \0";

                alt_up_pixel_buffer_dma_draw_box(pvga, 20, 161, 275, 225, 0x0000, 0);

                //Dibuja Cuadricula para Gráfica en el Tiempo
                alt_up_pixel_buffer_dma_draw_rectangle(pvga, 20, 161, 275, 225, 0x1863, 0);
                for (i = 1; i <= 3; i++)
                        alt_up_pixel_buffer_dma_draw_hline(pvga, 20, 275, 161 + (i*16), 0x1863, 0);
                for (i = 1; i <= 15; i++)
                        alt_up_pixel_buffer_dma_draw_vline(pvga, 20 + (i*16), 161, 225, 0x1863, 0);
                for (i = 0; i < 4; i++)
                        alt_up_pixel_buffer_dma_draw_vline(pvga, 20 + (i*64), 225, 227, 0xFFFF, 0);

                //Grafica Magnitud de la Señal en el Tiempo
                alt_up_char_buffer_string (cvga, "1", 3, 40);
                alt_up_char_buffer_string (cvga, "0", 3, 48);
                alt_up_char_buffer_string (cvga, "-1",3, 56);
```

```
		//Dibuja la Señal en el Tiempo
		alt_up_char_buffer_string (cvga, limpia, 4, 57);
		i = 20; k = 0;
		for (j = 0; j < 255; j++)
		{
			alt_up_pixel_buffer_dma_draw_line(pvga, i, r[zm*j]+193, i+1, r[zm*j+1]+193, 0x07E0, 0);
			k++;
			alt_up_pixel_buffer_dma_draw_line(pvga, i+1, r[zm*j+1]+193, i+1, r[zm*k]+193, 0x07E0, 0);
			i++;
			if ((j % 64) == 0)
			{
				fx1 = 1000 * (zm * (j *((float)1 / (float)(64 * fme))));
				sprintf(cx1, "%1.2f", fx1);
				alt_up_char_buffer_string (cvga, cx1, (4 + (16*(j/64))) , 57);
			}
		}
	}

/*****************************************************************************
 * Subroutine to write a pixel
*****************************************************************************/
	void write_pixel (int x, int y, short colour)
	{
		volatile short *vga_addr = (volatile short*)(0x08000000 + (y<<10) + (x<<1));
		*vga_addr = colour;
	}

/*****************************************************************************
 * Subroutine to send a string of text to the VGA monitor
*****************************************************************************/
	void VGA_text (int x, int y, char * text_ptr)
	{
		int offset;
		volatile char * character_buffer = (char *) 0x09000000; // VGA character buffer

		/* assume that the text string fits on one line */
		offset = (y << 7) + x;
		while ( *(text_ptr) )
		{
			*(character_buffer + offset) = *(text_ptr);	// write to the character buffer
			++text_ptr;
			++offset;
		}
	}

/*****************************************************************************
 * Draw a filled rectangle on the VGA monitor
*****************************************************************************/
	void VGA_box(int x1, int y1, int x2, int y2, short pixel_color)
	{
		int offset, row, col;
		volatile short * pixel_buffer = (short *) 0x08000000;	// VGA pixel buffer

		/* assume that the box coordinates are valid */
		for (row = y1; row <= y2; row++)
		{
			col = x1;
			while (col <= x2)
			{
				offset = (row << 9) + col;
				*(pixel_buffer + offset) = pixel_color;		// compute halfword address, set pixel
```

```
                                        ++col;
                                }
                        }
                }

/*****************************************************************************
 * Fast Fourier Transform
*****************************************************************************/
                void fft(int n, int m, double x[], double y[])
                {
                        int i,j,k,n1,n2;
                        double c,s,e,a,t1,t2;

                        j = 0; /* bit-reverse */
                        n2 = n/2;
                        for (i = 1; i < n - 1; i++)
                        {
                                n1 = n2;
                                while (j >= n1)
                                {
                                        j = j - n1;
                                        n1 = n1/2;
                                }
                                j = j + n1;

                                if (i < j)
                                {
                                        t1 = x[i];
                                        x[i] = x[j];
                                        x[j] = t1;
                                        t1 = y[i];
                                        y[i] = y[j];
                                        y[j] = t1;
                                }
                        }

                        n1 = 0; /* FFT */
                        n2 = 1;

                        for (i = 0; i < m; i++)
                        {
                                n1 = n2;
                                n2 = n2 + n2;
                                e = -6.283185307179586 / n2;
                                a = 0.0;

                                for (j = 0; j < n1; j++)
                                {
                                        c = cos(a);
                                        s = sin(a);
                                        a = a + e;

                                        for (k = j; k < n; k = k + n2)
                                        {
                                                t1 = c*x[k+n1] - s*y[k+n1];
                                                t2 = s*x[k+n1] + c*y[k+n1];
                                                x[k+n1] = x[k] - t1;
                                                y[k+n1] = y[k] - t2;
                                                x[k] = x[k] + t1;
                                                y[k] = y[k] + t2;
                                        }
                                }
```

```
		}
	}

/*****************************************************************************************
 * Magnitude
*****************************************************************************************/
	void magnitud (int n, double x[], double y[], double r[])
	{
		int i;
		double n1, n2;

		for (i = 0; i < n; i++)
		{
			n1 = pow(x[i],2);
			n2 = pow(y[i],2);
			r[i] = sqrt(n1 + n2);
		}
	}

/*****************************************************************************************
 * Integer transform and rounded
*****************************************************************************************/
	void entero (int n, double m[], int r[])
	{
		int i, temp;
		double t;

		for (i = 0; i < n; i++)
		{
			t = m[i]*10;
			temp = (int) t % 10;
			if (temp >= 5)
				r[i] = ((int) m[i]) + 1;
			else
				r[i] = (int) m[i];
		}
	}

/*****************************************************************************************
 * Max of an array and his position
*****************************************************************************************/
	int position (int n, int m[])
	{
		int i, pos, max = 0;
		for (i = 0; i < n; i++)
		{
			if (m[i] > max)
			{
				max = m[i];
				pos = i;
			}
		}
		return pos;
	}

	int maximo (int n, int m[])
	{
		int i, max = 0;
		for (i = 0; i < n; i++)
		{
			if (m[i] > max)
				max = m[i];
```

```
		}
		return max;
	}

/*****************************************************************************************
 * Get a character from the RS232 UART
*****************************************************************************************/
	int getcharRS232()
	{
		unsigned int rs232word;

		do
			rs232word = *pRS232;
		while ( rs232bytes_available(rs232word) == 0 );

		return rs232word & 0xFF;
	}


/*****************************************************************************************
 * FFT2
*****************************************************************************************/
	void fft2 (int n, int m, double x[], double y[])
	{
		long i, i1, j, k, i2, l, l1, l2;
		double c1, c2, tx, ty, t1, t2, u1, u2, z;

		/* Do the bit reversal */
		i2 = n >> 1;
		j = 0;
		for (i = 0; i < n-1; i++)
		{
			if (i < j)
			{
				tx = x[i];
				ty = y[i];
				x[i] = x[j];
				y[i] = y[j];
				x[j] = tx;
				y[j] = ty;
			}
			k = i2;
			while (k <= j)
			{
				j -= k;
				k >>= 1;
			}
			j += k;
		}

		/* Compute the FFT */
		c1 = -1.0;
		c2 = 0.0;
		l2 = 1;
		for (l = 0; l < m; l++)
		{
			l1 = l2;
			l2 <<= 1;
			u1 = 1.0;
			u2 = 0.0;
			for (j = 0; j < l1; j++)
			{
```

```
                    for (i = j; i < n; i+=l2)
                    {
                        i1 = i + l1;
                        t1 = u1 * x[i1] - u2 * y[i1];
                        t2 = u1 * y[i1] + u2 * x[i1];
                        x[i1] = x[i] - t1;
                        y[i1] = y[i] - t2;
                        x[i] += t1;
                        y[i] += t2;
                    }
                    z =  u1 * c1 - u2 * c2;
                    u2 = u1 * c2 + u2 * c1;
                    u1 = z;
                }
                c2 = sqrt((1.0 - c1) / 2.0);
                c2 = -c2;
                c1 = sqrt((1.0 + c1) / 2.0);
            }

            /* Scaling for forward transform */
            //for (i = 0; i < n; i++)
            //{
                //x[i] /= n;
                //y[i] /= n;
            //}
}
```

ANEXO B

CODIGO DEL GENERADOR DE SEÑALES EN MATLAB

```
function varargout = Generador(varargin)

% GENERADOR M-file for Generador.fig
%      GENERADOR, by itself, creates a new GENERADOR or raises the existing
%      singleton*.
%
%      H = GENERADOR returns the handle to a new GENERADOR or the handle to
%      the existing singleton*.
%
%      GENERADOR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GENERADOR.M with the given input arguments.
%
%      GENERADOR('Property','Value',...) creates a new GENERADOR or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Generador_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Generador_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help Generador
% Last Modified by GUIDE v2.5 26-May-2012 13:34:30
% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Generador_OpeningFcn, ...
                   'gui_OutputFcn',  @Generador_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);

if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
        gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Generador is made visible.
function Generador_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Generador (see VARARGIN)
handles.ejex = 0:pi / 360:2*pi;
% Choose default command line output for Generador
```

```
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Generador wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Generador_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
fun = get(handles.popupmenu1,'Value');

f = 1000*get(handles.frecuencia,'Value');
if (f == 0)
        f = 1;
end

fg = 2000*get(handles.rango,'Value');
%fg_1 = 6000 * get(handles.rango,'Value');
handles.cic = 20 * get(handles.ciclos,'Value');
if (handles.cic == 0)
        handles.cic = 0.1;
end

handles.ejex1 = 0:1/(64*f):64/f;

Ts1 = 1/(64*f);

f_m = 1000 * get(handles.frec_m,'Value');
if (f_m == 0)
        f_m = 1;
end

f_c = 1000 * get(handles.frec_c,'Value');
if (f_c == 0)
        f_c = 1;
end

handles.ejex2 = 0:1/(64*f_c):64/f_c;
Ts = 1/(64*f_c);
Fs = 1/Ts;

switch fun
        case 1
                y1 = sin(2*pi*f*handles.ejex1);
                axes(handles.grafica);
                dibujarGrafica(handles.ejex1,1,handles.cic,y1,f);  %Dibuja la Señal en el dominio del Tiempo.
```

```
                title('SEÑAL GENERADA EN EL TIEMPO');
                axes(handles.espectro);
                plotspec(y1,Ts1,(2*f),fun);          %Dibuja la Señal en el dominio de la Frecuencia.
                set(handles.text11,'String',(2*f));  %Escribe el valor de Slider en static text
                set(handles.rango,'Value',((2*f)/2000));
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
                        zoom off;
                        set(handles.zoom,'String','ZOOM OFF');
                end
                %*-*-*-*-*-*-

        case 2
                y2 = cos(2*pi*f*handles.ejex1);
                axes(handles.grafica);
                dibujarGrafica(handles.ejex1,1,handles.cic,y2,f);  %Dibuja la Señal en el dominio del Tiempo.
                title('SEÑAL GENERADA EN EL TIEMPO');
                axes(handles.espectro);
                plotspec(y2,Ts1,(2*f),fun);  %Dibuja la Señal en el dominio de la Frecuencia.
                set(handles.text11,'String',(2*f));   %Escribe el valor de Slider en static text
                set(handles.rango,'Value',((2*f)/2000));
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
                        zoom off;
                        set(handles.zoom,'String','ZOOM OFF');
                end
                %*-*-*-*-*-*-

        case 3
                y3 = square(2*pi*f*handles.ejex1);
                axes(handles.grafica);
                dibujarGrafica(handles.ejex1,1.5,handles.cic,y3,f);  %Dibuja la Señal en el dominio del Tiempo.
                title('SEÑAL GENERADA EN EL TIEMPO');
                axes(handles.espectro);
                plotspec(y3,Ts1,fg,fun);   %Dibuja la Señal en el dominio de la Frecuencia.
                n = floor(log(length(y3))/log(2));
                N = 2^n;
                ssf = (-N/2:N/2-1)/(Ts1*N);
                set(handles.text11,'String',round(max(ssf)));
                set(handles.rango,'Value',1);
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
                        zoom off;
                        set(handles.zoom,'String','ZOOM OFF');
                end
                %*-*-*-*-*-*-

        case 4
                y4 = sawtooth(2*pi*f*handles.ejex1);
                axes(handles.grafica);
```

```
        dibujarGrafica(handles.ejex1,(1.5),handles.cic,y4,f);   %Dibuja la Señal en el dominio del Tiempo.
        title('SEÑAL GENERADA EN EL TIEMPO');
        axes(handles.espectro);
        plotspec(y4,Ts1,fg,fun);    %Dibuja la Señal en el dominio de la Frecuencia.
        n = floor(log(length(y4))/log(2));
        N = 2^n;
        ssf = (-N/2:N/2-1)/(Ts1*N);
        set(handles.text11,'String',round(max(ssf)));
        set(handles.rango,'Value',1);
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 5
        y_m = cos(2*pi*f_m*handles.ejex2) + 1;
        axes(handles.grafica);
        %Dibuja la Señal en el dominio del Tiempo.
        plot(handles.ejex2,y_m,'--','LineWidth',1,'Color',[1 0 0]); hold on;
        plot(handles.ejex2,-y_m,'--','LineWidth',1,'Color',[1 0 0]); hold on;
        y_c = cos(2*pi*f_c*handles.ejex2);
        y_am = y_m.*y_c;
        plot(handles.ejex2,y_am,'LineWidth',2,'color',[0 1 0])
        hold off;
        set(gca,'Color',[0.16 0.16 0.16]);
        title('SEÑAL AM');
        xlabel('Tiempo (s)')
        ylabel('Magnitud (V)')
        grid on;
        axis([0 (handles.cic/f_m) -2.5 2.5]);
        axes(handles.espectro);
        plotspec(y_am,Ts,2.5*f_c,fun);    %Dibuja la Señal en el dominio de la Frecuencia.
        set(handles.text11,'String',2.5*f_c);
        set(handles.rango,'Value',1);
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 6
        x_m1 = cos(2*pi*f_m*handles.ejex2);
        y_fm = modulate(x_m1,f_c,Fs,'fm');
        axes(handles.grafica);
        %Dibuja la Señal en el dominio del Tiempo.
        dibujarGrafica(handles.ejex2,1.5,handles.cic,y_fm,f_m);   %Dibuja la Señal en el dominio del Tiempo.
        title('SEÑAL FM');
        xlabel('Tiempo (s)')
        ylabel('Magnitud (V)')
        grid on;
        axes(handles.espectro);
```

```
                    plotspec(y_fm,Ts,4800,1);    %Dibuja la Señal en el dominio de la Frecuencia.
                    n = floor(log(length(y_fm))/log(2));
                    N = 2^n;
                    ssf = (-N/2:N/2-1)/(Ts*N);
                    set(handles.text11,'String',4800);
                    set(handles.rango,'Value',1);
                    title('ESPECTRO');
                    a = get(handles.zoom,'Value');
                    if a == 1
                            zoom on;
                            set(handles.zoom,'String','ZOOM ON');
                    else
                            zoom off;
                            set(handles.zoom,'String','ZOOM OFF');
                    end
                    %*-*-*-*-*-*-

end
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end

function plotspec(w,Ts,fg,fun1)

n = floor(log(length(w))/log(2));
N = 2^n;
fw = abs(fft (w(1:N)));
ssf = (-N/2:N/2-1)/(Ts*N);
if ((fun1 == 3) || (fun1 == 4) || (fun1 == 6))
        fg = max(ssf);
end

fws = fftshift(fw);
plot(ssf,fws,'color',[0 1 1]);
set(gca,'Color',[0.16 0.16 0.16]);
xlabel('Frequencia')
ylabel('Magnitud')
axis([-fg fg 0 (max(fws)+100)]);

function dibujarGrafica(ejex,amplitud,ciclos,y,f)
max = amplitud;
min = (-1)*(max);
plot(ejex,y,'LineWidth',2,'color',[0 1 0]);
set(gca,'Color',[0.16 0.16 0.16]);
xlabel('Tiempo')
ylabel('Magnitud')
grid on;
axis([0 (ciclos/f) min max]);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

fun = get(handles.popupmenu1,'Value');

f = 1000*get(handles.frecuencia,'Value');
if (f == 0)
        f = 1;
end

fg = 2000*get(handles.rango,'Value');
%fg_1 = 6000 * get(handles.rango,'Value');
handles.cic = 20 * get(handles.ciclos,'Value');
if (handles.cic == 0)
        handles.cic = 0.1;
end

handles.ejex1 = 0:1/(64*f):64/f;

Ts1 = 1/(64*f);

f_m = 1000 * get(handles.frec_m,'Value');
if (f_m == 0)
        f_m = 1;
end

f_c = 1000 * get(handles.frec_c,'Value');
if (f_c == 0)
        f_c = 1;
end

handles.ejex2 = 0:1/(64*f_c):64/f_c;
Ts = 1/(64*f_c);
Fs = 1/Ts;

com = get(handles.popupmenu3,'Value');
switch com
        case 1
                puerto = 'COM1';
        case 2
                puerto = 'COM2';
        case 3
                puerto = 'COM3';
        case 4
                puerto = 'COM4';
        case 5
                puerto = 'COM5';
        case 6
                puerto = 'COM6';
        case 7
                puerto = 'COM7';
        case 8
                puerto = 'COM8';
        case 9
                puerto = 'COM9';
        case 10
                puerto = 'COM10';
        case 11
                puerto = 'COM11';
        case 12
                puerto = 'COM12';
end
```

```
switch fun
        case 1
                y1 = sin(2*pi*f*handles.ejex1);
                y = 0.5 + (0.5*sin(2*pi*f*handles.ejex1));
                axes(handles.grafica);
                dibujarGrafica(handles.ejex1,1,handles.cic,y1,f);   %Dibuja la Señal en el dominio del Tiempo.
                title('SEÑAL GENERADA EN EL TIEMPO');
                axes(handles.espectro);
                plotspec(y1,Ts1,(2*f),fun);   %Dibuja la Señal en el dominio de la Frecuencia.
                set(handles.text11,'String',(2*f));  %Escribe el valor de Slider en static text
                set(handles.rango,'Value',((2*f)/2000));
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
                        zoom off;
                        set(handles.zoom,'String','ZOOM OFF');
                end

                SerDE2 = serial(puerto);
                set(SerDE2,'BaudRate',115200);
                set(SerDE2,'DataBits',8);
                set(SerDE2,'Parity','odd');
                set(SerDE2,'StopBits',1);
                set(SerDE2,'FlowControl','none');

                try
                        fopen(SerDE2);
                        %*-*-*-*-*-*-

                        for k = 1:1024

                                j = 1;
                                g = 0;

                                for j = 1:4096
                                        if (((j-1)/4096) <= y(k) <= (j/4096))
                                                dato(k) = y(k) * 4095;
                                                g = 1;
                                        end
                                        if g == 0
                                                continue
                                        else
                                                break
                                        end
                                end

                                datoent(k) = round(dato(k));
                                datobin = dec2bin(datoent(k),12);
                                datoh(k) = bin2dec(datobin(1:5));

                                if (datoh(k) == 0)
                                        fprintf(SerDE2,'%c', char(160));
                                else
                                        fprintf(SerDE2,'%c', char(datoh(k)));
                                end

                                datol(k) = bin2dec(datobin(6:12));
```

```
                        if (datol(k) == 0)
                                fprintf(SerDE2,'%c', char(160));
                        else
                                fprintf(SerDE2,'%c', char(datol(k)));
                        end

                end

                f_mt = round(f);
                f_mtbin = dec2bin(f_mt,10);
                f_mt = bin2dec(f_mtbin(1:7));
                if (f_mt == 0)
                        fprintf(SerDE2,'%c', char(160));
                else
                        fprintf(SerDE2,'%c', char(f_mt));
                end

                f_mt = bin2dec(f_mtbin(8:10));
                if (f_mt == 0)
                        fprintf(SerDE2,'%c', char(160));
                else
                        fprintf(SerDE2,'%c', char(f_mt));
                end

                %CERRAR el puerto COM al finalizar
                fclose(SerDE2);
                delete(SerDE2)
                clear SerDE2
                disp('STOP')
        catch
                errordlg({'EL PUERTO COM SELECCIONADO NO ES EL CORRECTO!','SELECCIONE OTRO PUERTO
DE COMUNICACIÓN SERIAL'},'ERROR');
        end

        %*-*-*-*-*-*-
case 2
        y2 = cos(2*pi*f*handles.ejex1);
        y = 0.5+0.5*y2;
        axes(handles.grafica);
        dibujarGrafica(handles.ejex1,1,handles.cic,y2,f);   %Dibuja la Señal en el dominio del Tiempo.
        title('SEÑAL GENERADA EN EL TIEMPO');
        axes(handles.espectro);
        plotspec(y2,Ts1,(2*f),fun);    %Dibuja la Señal en el dominio de la Frecuencia.
        set(handles.text11,'String',(2*f));    %Escribe el valor de Slider en static text
        set(handles.rango,'Value',((2*f)/2000));
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end

        SerDE2 = serial(puerto);
        set(SerDE2,'BaudRate',115200);
        set(SerDE2,'DataBits',8);
        set(SerDE2,'Parity','odd');
        set(SerDE2,'StopBits',1);
        set(SerDE2,'FlowControl','none');
```

```
try
        fopen(SerDE2);
        %*-*-*-*-*-*-

        for k = 1:1024

        j = 1;
        g = 0;

                for j = 1:4096
                        if (((j-1)/4096) <= y(k) <= (j/4096))
                                dato(k) = y(k) * 4095;
                                g = 1;
                        end
                        if g == 0
                                continue
                        else
                                break
                        end
                end

                datoent(k) = round(dato(k));
                datobin = dec2bin(datoent(k),12);
                datoh(k) = bin2dec(datobin(1:5));

                if (datoh(k) == 0)
                        fprintf(SerDE2,'%c', char(160));
                else
                        fprintf(SerDE2,'%c', char(datoh(k)));
                end

                datol(k) = bin2dec(datobin(6:12));

                if (datol(k) == 0)
                        fprintf(SerDE2,'%c', char(160));
                else
                        fprintf(SerDE2,'%c', char(datol(k)));
                end

        end

        f_mt = round(f);
        f_mtbin = dec2bin(f_mt,10);
        f_mt = bin2dec(f_mtbin(1:7));
        if (f_mt == 0)
                fprintf(SerDE2,'%c', char(160));
        else
                fprintf(SerDE2,'%c', char(f_mt));
        end

        f_mt = bin2dec(f_mtbin(8:10));
        if (f_mt == 0)
                fprintf(SerDE2,'%c', char(160));
        else
                fprintf(SerDE2,'%c', char(f_mt));
        end

        %CERRAR el puerto COM al finalizar
        fclose(SerDE2);
        delete(SerDE2)
        clear SerDE2
        disp('STOP')
```

```
                catch
                        errordlg({'EL PUERTO COM SELECCIONADO NO ES EL CORRECTO!','SELECCIONE OTRO PUERTO
DE COMUNICACIÓN SERIAL'},'ERROR');
                end
                %*-*-*-*-*-*-

        case 3
                y3 = square(2*pi*f*handles.ejex1);
                y = 0.5 +(0.5* y3);
                axes(handles.grafica);
                dibujarGrafica(handles.ejex1,1.5,handles.cic,y3,f);        %Dibuja la Señal en el dominio del Tiempo.
                title('SEÑAL GENERADA EN EL TIEMPO');
                axes(handles.espectro);
                plotspec(y3,Ts1,fg,fun);            %Dibuja la Señal en el dominio de la Frecuencia.
                n = floor(log(length(y3))/log(2));
                N = 2^n;
                ssf = (-N/2:N/2-1)/(Ts1*N);
                set(handles.text11,'String',round(max(ssf)));
                set(handles.rango,'Value',1);
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
                        zoom off;
                        set(handles.zoom,'String','ZOOM OFF');
                end

                SerDE2 = serial(puerto);
                set(SerDE2,'BaudRate',115200);
                set(SerDE2,'DataBits',8);
                set(SerDE2,'Parity','odd');
                set(SerDE2,'StopBits',1);
                set(SerDE2,'FlowControl','none');

                try
                        fopen(SerDE2);
                        %*-*-*-*-*-*-

                        for k = 1:1024
                                j = 1;
                                g = 0;

                                for j = 1:4096
                                        if (((j-1)/4096) <= y(k) <= (j/4096))
                                                dato(k) = y(k) * 4095;
                                                g = 1;
                                        end
                                        if g == 0
                                                continue
                                        else
                                                break
                                        end
                                end

                                datoent(k) = round(dato(k));
                                datobin = dec2bin(datoent(k),12);
                                datoh(k) = bin2dec(datobin(1:5));

                                if (datoh(k) == 0)
```

```
                                        fprintf(SerDE2,'%c', char(160));
                                else
                                        fprintf(SerDE2,'%c', char(datoh(k)));
                                end

                                datol(k) = bin2dec(datobin(6:12));

                                if (datol(k) == 0)
                                        fprintf(SerDE2,'%c', char(160));
                                else
                                        fprintf(SerDE2,'%c', char(datol(k)));
                                end

                        end

                        f_mt = round(f);
                        f_mtbin = dec2bin(f_mt,10);
                        f_mt = bin2dec(f_mtbin(1:7));
                        if (f_mt == 0)
                                fprintf(SerDE2,'%c', char(160));
                        else
                                fprintf(SerDE2,'%c', char(f_mt));
                        end

                        f_mt = bin2dec(f_mtbin(8:10));
                        if (f_mt == 0)
                                fprintf(SerDE2,'%c', char(160));
                        else
                                fprintf(SerDE2,'%c', char(f_mt));
                        end

                        %CERRAR el puerto COM al finalizar
                        fclose(SerDE2);
                        delete(SerDE2)
                        clear SerDE2
                        disp('STOP')

                catch
                        errordlg({'EL PUERTO COM SELECCIONADO NO ES EL CORRECTO!','SELECCIONE OTRO PUERTO
DE COMUNICACIÓN SERIAL'},'ERROR');
                end
                %*-*-*-*-*-*-

        case 4
                y4 = sawtooth(2*pi*f*handles.ejex1);
                y = 0.5 + 0.5*y4;
                axes(handles.grafica);
                dibujarGrafica(handles.ejex1,(1.5),handles.cic,y4,f);    %Dibuja la Señal en el dominio del Tiempo.
                title('SEÑAL GENERADA EN EL TIEMPO');
                axes(handles.espectro);
                plotspec(y4,Ts1,fg,fun);    %Dibuja la Señal en el dominio de la Frecuencia.
                n = floor(log(length(y4))/log(2));
                N = 2^n;
                ssf = (-N/2:N/2-1)/(Ts1*N);
                set(handles.text11,'String',round(max(ssf)));
                set(handles.rango,'Value',1);
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
```

```
            zoom off;
            set(handles.zoom,'String','ZOOM OFF');
end

SerDE2 = serial(puerto);
set(SerDE2,'BaudRate',115200);
set(SerDE2,'DataBits',8);
set(SerDE2,'Parity','odd');
set(SerDE2,'StopBits',1);
set(SerDE2,'FlowControl','none');

try
            fopen(SerDE2);
            %*-*-*-*-*-*-

            for k = 1:1024
                        j = 1;
                        g = 0;

                        for j = 1:4096
                                    if (((j-1)/4096) <= y(k) <= (j/4096))
                                                dato(k) = y(k) * 4095;
                                                g = 1;
                                    end
                                    if g == 0
                                                continue
                                    else
                                                break
                                    end
                        end

                        datoent(k) = round(dato(k));
                        datobin = dec2bin(datoent(k),12);
                        datoh(k) = bin2dec(datobin(1:5));

                        if (datoh(k) == 0)
                                    fprintf(SerDE2,'%c', char(160));
                        else
                                    fprintf(SerDE2,'%c', char(datoh(k)));
                        end

                        datol(k) = bin2dec(datobin(6:12));

                        if (datol(k) == 0)
                                    fprintf(SerDE2,'%c', char(160));
                        else
                                    fprintf(SerDE2,'%c', char(datol(k)));
                        end
            end

            f_mt = round(f);
            f_mtbin = dec2bin(f_mt,10);
            f_mt = bin2dec(f_mtbin(1:7));
            if (f_mt == 0)
                        fprintf(SerDE2,'%c', char(160));
            else
                        fprintf(SerDE2,'%c', char(f_mt));
            end

            f_mt = bin2dec(f_mtbin(8:10));
            if (f_mt == 0)
                        fprintf(SerDE2,'%c', char(160));
```

```
                else
                        fprintf(SerDE2,'%c', char(f_mt));
                end

                %CERRAR el puerto COM al finalizar
                fclose(SerDE2);
                delete(SerDE2)
                clear SerDE2
                disp('STOP')
        catch
                errordlg({'EL PUERTO COM SELECCIONADO NO ES EL CORRECTO!','SELECCIONE OTRO PUERTO
DE COMUNICACIÓN SERIAL'},'ERROR');
        end
        %*-*-*-*-*-*-

case 5
        y_m = cos(2*pi*f_m*handles.ejex2) + 1;
        axes(handles.grafica);
        %Dibuja la Señal en el dominio del Tiempo.
        plot(handles.ejex2,y_m,'--','LineWidth',1,'Color',[1 0 0]);             hold on;
        plot(handles.ejex2,-y_m,'--','LineWidth',1,'Color',[1 0 0]);            hold on;
        y_c = cos(2*pi*f_c*handles.ejex2);
        y_am = y_m.*y_c;
        y = 0.5 + 0.25*y_am;
        plot(handles.ejex2,y_am,'LineWidth',2,'color',[0 1 0])
        hold off;
        set(gca,'Color',[0.16 0.16 0.16]);
        title('SEÑAL AM');
        xlabel('Tiempo (s)')
        ylabel('Magnitud (V)')
        grid on;
        axis([0 (handles.cic/f_m) -2.5 2.5]);
        axes(handles.espectro);
        plotspec(y_am,Ts,2.5*f_c,fun);          %Dibuja la Señal en el dominio de la Frecuencia.
        set(handles.text11,'String',2.5*f_c);
        set(handles.rango,'Value',1);
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end

        SerDE2 = serial(puerto);
        set(SerDE2,'BaudRate',115200);
        set(SerDE2,'DataBits',8);
        set(SerDE2,'Parity','odd');
        set(SerDE2,'StopBits',1);
        set(SerDE2,'FlowControl','none');

        try
                fopen(SerDE2);
                %*-*-*-*-*-*-

                for k = 1:1024
                        j = 1;
                        g = 0;

                        for j = 1:4096
```

```
                        if (((j-1)/4096) <= y(k) <= (j/4096))
                                dato(k) = y(k) * 4095;
                                g = 1;
                        end
                        if g == 0
                                continue
                        else
                                break
                        end
                end

                datoent(k) = round(dato(k));
                datobin = dec2bin(datoent(k),12);
                datoh(k) = bin2dec(datobin(1:5));

                if (datoh(k) == 0)
                        fprintf(SerDE2,'%c', char(160));
                else
                        fprintf(SerDE2,'%c', char(datoh(k)));
                end

                datol(k) = bin2dec(datobin(6:12));

                if (datol(k) == 0)
                        fprintf(SerDE2,'%c', char(160));
                else
                        fprintf(SerDE2,'%c', char(datol(k)));
                end

        end

        f_mt = round(f_c);
        f_mtbin = dec2bin(f_mt,10);
        f_mt = bin2dec(f_mtbin(1:7));
        if (f_mt == 0)
                fprintf(SerDE2,'%c', char(160));
        else
                fprintf(SerDE2,'%c', char(f_mt));
        end

        f_mt = bin2dec(f_mtbin(8:10));
        if (f_mt == 0)
                fprintf(SerDE2,'%c', char(160));
        else
                fprintf(SerDE2,'%c', char(f_mt));
        end

        %CERRAR el puerto COM al finalizar
        fclose(SerDE2);
        delete(SerDE2)
        clear SerDE2
        disp('STOP')

    catch
        errordlg({'EL PUERTO COM SELECCIONADO NO ES EL CORRECTO!','SELECCIONE OTRO PUERTO
DE COMUNICACIÓN SERIAL'},'ERROR');
    end
    %*-*-*-*-*-*-

case 6
    x_m1 = cos(2*pi*f_m*handles.ejex2);
    y_fm = modulate(x_m1,f_c,Fs,'fm');
```

```
y = 0.5+0.5*y_fm;
axes(handles.grafica);
%Dibuja la Señal en el dominio del Tiempo.
dibujarGrafica(handles.ejex2,1.5,handles.cic,y_fm,f_m);          %Dibuja la Señal en el dominio del Tiempo.
title('SEÑAL FM');
xlabel('Tiempo (s)')
ylabel('Magnitud (V)')
grid on;
axes(handles.espectro);
plotspec(y_fm,Ts,4800,1);        %Dibuja la Señal en el dominio de la Frecuencia.
n = floor(log(length(y_fm))/log(2));
N = 2^n;
ssf = (-N/2:N/2-1)/(Ts*N);
set(handles.text11,'String',4800);
set(handles.rango,'Value',1);
title('ESPECTRO');
a = get(handles.zoom,'Value');
if a == 1
        zoom on;
        set(handles.zoom,'String','ZOOM ON');
else
        zoom off;
        set(handles.zoom,'String','ZOOM OFF');
end

SerDE2 = serial(puerto);
set(SerDE2,'BaudRate',115200);
set(SerDE2,'DataBits',8);
set(SerDE2,'Parity','odd');
set(SerDE2,'StopBits',1);
set(SerDE2,'FlowControl','none');

try
        fopen(SerDE2);
        %*-*-*-*-*-*-
        for k = 1:1024
                j = 1;
                g = 0;

                for j = 1:4096
                        if (((j-1)/4096) <= y(k) <= (j/4096))
                                dato(k) = y(k) * 4095;
                                g = 1;
                        end
                        if g == 0
                                continue
                        else
                                break
                        end
                end

                datoent(k) = round(dato(k));
                datobin = dec2bin(datoent(k),12);
                datoh(k) = bin2dec(datobin(1:5));

                if (datoh(k) == 0)
                        fprintf(SerDE2,'%c', char(160));
                else
                        fprintf(SerDE2,'%c', char(datoh(k)));
                end
```

```
                                        datol(k) = bin2dec(datobin(6:12));

                                        if (datol(k) == 0)
                                                fprintf(SerDE2,'%c', char(160));
                                        else
                                                fprintf(SerDE2,'%c', char(datol(k)));
                                        end

                                end

                                f_mt = round(f_c);
                                f_mtbin = dec2bin(f_mt,10);
                                f_mt = bin2dec(f_mtbin(1:7));
                                if (f_mt == 0)
                                        fprintf(SerDE2,'%c', char(160));
                                else
                                        fprintf(SerDE2,'%c', char(f_mt));
                                end

                                f_mt = bin2dec(f_mtbin(8:10));
                                if (f_mt == 0)
                                        fprintf(SerDE2,'%c', char(160));
                                else
                                        fprintf(SerDE2,'%c', char(f_mt));
                                end

                                %CERRAR el puerto COM al finalizar
                                fclose(SerDE2);
                                delete(SerDE2)
                                clear SerDE2
                                disp('STOP')

                        catch
                                errordlg({'EL PUERTO COM SELECCIONADO NO ES EL CORRECTO!','SELECCIONE OTRO PUERTO
        DE COMUNICACIÓN SERIAL'},'ERROR');
                        end
                        %*-*-*-*-*-*-

        end

guidata(hObject,handles)

function frecuencia_Callback(hObject, eventdata, handles)
% hObject    handle to frecuencia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of frecuencia as text
%       str2double(get(hObject,'String')) returns contents of frecuencia as a double
handles.f = get(hObject,'Value');                       %Carga en handles.slider1 el valor del Slider
handles.f = 1000.*handles.f;
if (handles.f == 0)
        handles.f = 1;
end
set(handles.text8,'String',handles.f);          %Escribe el valor de Slider en static text

fun = get(handles.popupmenu1,'Value');

f = handles.f;

fg = 2000*get(handles.rango,'Value');
```

```
handles.cic = 20 * get(handles.ciclos,'Value');
if (handles.cic == 0)
        handles.cic = 0.1;
end

handles.ejex1 = 0:1/(64*f):64/f;

Ts1 = 1/(64*f);

f_m = 1000 * get(handles.frec_m,'Value');

f_c = 1000 * get(handles.frec_c,'Value');

handles.ejex2 = 0:1/(64*f_c):64/f_c;
Ts = 1/(64*f_c);
Fs = 1/Ts;

switch fun
    case 1
        y1 = sin(2*pi*f*handles.ejex1);
        axes(handles.grafica);
        dibujarGrafica(handles.ejex1,1,handles.cic,y1,f);      %Dibuja la Señal en el dominio del Tiempo.
        title('SEÑAL GENERADA EN EL TIEMPO');
        axes(handles.espectro);
        plotspec(y1,Ts1,(2*f),fun);      %Dibuja la Señal en el dominio de la Frecuencia.
        set(handles.text11,'String',(2*f));          %Escribe el valor de Slider en static text
        set(handles.rango,'Value',((2*f)/2000));
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
            zoom on;
            set(handles.zoom,'String','ZOOM ON');
        else
            zoom off;
            set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

    case 2
        y2 = cos(2*pi*f*handles.ejex1);
        axes(handles.grafica);
        dibujarGrafica(handles.ejex1,1,handles.cic,y2,f);      %Dibuja la Señal en el dominio del Tiempo.
        title('SEÑAL GENERADA EN EL TIEMPO');
        axes(handles.espectro);
        plotspec(y2,Ts1,(2*f),fun);      %Dibuja la Señal en el dominio de la Frecuencia.
        set(handles.text11,'String',(2*f));          %Escribe el valor de Slider en static text
        set(handles.rango,'Value',((2*f)/2000));
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
            zoom on;
            set(handles.zoom,'String','ZOOM ON');
        else
            zoom off;
            set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

    case 3
        y3 = square(2*pi*f*handles.ejex1);
        axes(handles.grafica);
```

```
        dibujarGrafica(handles.ejex1,1.5,handles.cic,y3,f);       %Dibuja la Señal en el dominio del Tiempo.
        title('SEÑAL GENERADA EN EL TIEMPO');
        axes(handles.espectro);
        plotspec(y3,Ts1,fg,fun);              %Dibuja la Señal en el dominio de la Frecuencia.
        n = floor(log(length(y3))/log(2));
        N = 2^n;
        ssf = (-N/2:N/2-1)/(Ts1*N);
        set(handles.text11,'String',round(max(ssf)));
        set(handles.rango,'Value',1);
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 4
        y4 = sawtooth(2*pi*f*handles.ejex1);
        axes(handles.grafica);
        dibujarGrafica(handles.ejex1,(1.5),handles.cic,y4,f);                %Dibuja la Señal en el dominio del
Tiempo.
        title('SEÑAL GENERADA EN EL TIEMPO');
        axes(handles.espectro);
        plotspec(y4,Ts1,fg,fun);              %Dibuja la Señal en el dominio de la Frecuencia.
        n = floor(log(length(y4))/log(2));
        N = 2^n;
        ssf = (-N/2:N/2-1)/(Ts1*N);
        set(handles.text11,'String',round(max(ssf)));
        set(handles.rango,'Value',1);
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 5
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 6
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
```

```
                    set(handles.zoom,'String','ZOOM OFF');
                end
                %*-*-*-*-*-*-

end
guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function frecuencia_CreateFcn(hObject, eventdata, handles)
% hObject    handle to frecuencia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end

% --- Executes on slider movement.
function frec_m_Callback(hObject, eventdata, handles)
% hObject    handle to frec_m (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.f_m=get(hObject,'Value'); %Carga en handles.slider1 el valor del Slider
f_m = 1000.*handles.f_m;
if (f_m == 0)
        f_m = 1;
end
set(handles.v_f_m,'String',f_m);            %Escribe el valor de Slider en static text

handles.ran = get(hObject,'Value');         %Carga en handles.slider1 el valor del Slider
handles.ran = 2000.*handles.ran;
if (handles.ran == 0)
        handles.ran = 1.5;
end
%set(handles.text11,'String',handles.ran); %Escribe el valor de Slider en static text

fun = get(handles.popupmenu1,'Value');

f = 1000*get(handles.frecuencia,'Value');
if (f == 0)
        f = 1;
end

fg = handles.ran;

handles.cic = 20 * get(handles.ciclos,'Value');
if (handles.cic == 0)
        handles.cic = 0.1;
end

handles.ejex1 = 0:1/(64*f):64/f;

Ts1 = 1/(64*f);

f_c = 1000 * get(handles.frec_c,'Value');
if (f_c == 0)
        f_c = 1;
```

```
end

handles.ejex2 = 0:1/(64*f_c):64/f_c;
Ts = 1/(64*f_c);
Fs = 1/Ts;

switch fun
	case 1
		%*-*-*-*-*-*-
	case 2
		%*-*-*-*-*-*-
	case 3
		%*-*-*-*-*-*-
	case 4
		%*-*-*-*-*-*-
	case 5
		y_m = cos(2*pi*f_m*handles.ejex2) + 1;
		axes(handles.grafica);
		%Dibuja la Señal en el dominio del Tiempo.
		plot(handles.ejex2,y_m,'--','LineWidth',1,'Color',[1 0 0]);		hold on;
		plot(handles.ejex2,-y_m,'--','LineWidth',1,'Color',[1 0 0]);		hold on;
		y_c = cos(2*pi*f_c*handles.ejex2);
		y_am = y_m.*y_c;
		plot(handles.ejex2,y_am,'LineWidth',2,'color',[0 1 0])
		hold off;
		set(gca,'Color',[0.16 0.16 0.16]);
		title('SEÑAL AM');
		xlabel('Tiempo (s)')
		ylabel('Magnitud (V)')
		grid on;
		axis([0 (handles.cic/f_m) -2.5 2.5]);
		axes(handles.espectro);
		plotspec(y_am,Ts,2.5*f_c,fun);		%Dibuja la Señal en el dominio de la Frecuencia.
		set(handles.text11,'String',2.5*f_c);
		set(handles.rango,'Value',1);
		title('ESPECTRO');
		a = get(handles.zoom,'Value');
		if a == 1
			zoom on;
			set(handles.zoom,'String','ZOOM ON');
		else
			zoom off;
			set(handles.zoom,'String','ZOOM OFF');
		end
		%*-*-*-*-*-*-

	case 6
		x_m1 = cos(2*pi*f_m*handles.ejex2);
		y_fm = modulate(x_m1,f_c,Fs,'fm');
		axes(handles.grafica);
		%Dibuja la Señal en el dominio del Tiempo.
		dibujarGrafica(handles.ejex2,1.5,handles.cic,y_fm,f_m);		%Dibuja la Señal en el dominio del
	Tiempo.
		title('SEÑAL FM');
		xlabel('Tiempo (s)')
		ylabel('Magnitud (V)')
		grid on;
		axes(handles.espectro);
		plotspec(y_fm,Ts,4800,1);		%Dibuja la Señal en el dominio de la Frecuencia.
		n = floor(log(length(y_fm))/log(2));
		N = 2^n;
		ssf = (-N/2:N/2-1)/(Ts*N);
```

```
                    set(handles.text11,'String',4800);
                    set(handles.rango,'Value',1);
                    title('ESPECTRO');
                    a = get(handles.zoom,'Value');
                    if a == 1
                            zoom on;
                            set(handles.zoom,'String','ZOOM ON');
                    else
                            zoom off;
                            set(handles.zoom,'String','ZOOM OFF');
                    end
                    %*-*-*-*-*-*-

end


guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function frec_m_CreateFcn(hObject, eventdata, handles)
% hObject    handle to frec_m (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function frec_c_Callback(hObject, eventdata, handles)
% hObject    handle to frec_c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%       get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.f_c1 = get(hObject,'Value');            %Carga en handles.slider1 el valor del Slider
f_c = 1000.*handles.f_c1;
if (f_c == 0)
        f_c = 1;
end
set(handles.v_f_c,'String',f_c);                %Escribe el valor de Slider en static text

handles.ran = get(hObject,'Value');             %Carga en handles.slider1 el valor del Slider
handles.ran = 2000.*handles.ran;
if (handles.ran == 0)
        handles.ran = 1.5;
end
%set(handles.text11,'String',handles.ran);              %Escribe el valor de Slider en static text

fun = get(handles.popupmenu1,'Value');

f = 1000*get(handles.frecuencia,'Value');
if (f == 0)
        f = 1;
end

fg = handles.ran;

handles.cic = 20 * get(handles.ciclos,'Value');
if (handles.cic == 0)
```

```
		handles.cic = 0.1;
end

handles.ejex1 = 0:1/(64*f):64/f;

Ts1 = 1/(64*f);

f_m = 1000 * get(handles.frec_m,'Value');
if (f_m == 0)
		f_m = 1;
end

handles.ejex2 = 0:1/(64*f_c):64/f_c;
Ts = 1/(64*f_c);
Fs = 1/Ts;

switch fun
		case 1
			%*-*-*-*-*-*-
		case 2
			%*-*-*-*-*-*-
		case 3
			%*-*-*-*-*-*-
		case 4
			%*-*-*-*-*-*-
		case 5
			y_m = cos(2*pi*f_m*handles.ejex2) + 1;
			axes(handles.grafica);
			%Dibuja la Señal en el dominio del Tiempo.
			plot(handles.ejex2,y_m,'--','LineWidth',1,'Color',[1 0 0]);		hold on;
			plot(handles.ejex2,-y_m,'--','LineWidth',1,'Color',[1 0 0]);		hold on;
			y_c = cos(2*pi*f_c*handles.ejex2);
			y_am = y_m.*y_c;
			plot(handles.ejex2,y_am,'LineWidth',2,'color',[0 1 0])
			hold off;
			set(gca,'Color',[0.16 0.16 0.16]);
			title('SEÑAL AM');
			xlabel('Tiempo (s)')
			ylabel('Magnitud (V)')
			grid on;
			axis([0 (handles.cic/f_m) -2.5 2.5]);
			axes(handles.espectro);
			plotspec(y_am,Ts,2.5*f_c,fun);		%Dibuja la Señal en el dominio de la Frecuencia.
			set(handles.text11,'String',2.5*f_c);
			set(handles.rango,'Value',1);
			title('ESPECTRO');
			a = get(handles.zoom,'Value');
			if a == 1
				zoom on;
				set(handles.zoom,'String','ZOOM ON');
			else
				zoom off;
				set(handles.zoom,'String','ZOOM OFF');
			end
			%*-*-*-*-*-*-

		case 6
			x_m1 = cos(2*pi*f_m*handles.ejex2);
			y_fm = modulate(x_m1,f_c,Fs,'fm');
			axes(handles.grafica);
			%Dibuja la Señal en el dominio del Tiempo.
```

```
                dibujarGrafica(handles.ejex2,1.5,handles.cic,y_fm,f_m);          %Dibuja la Señal en el dominio del
        Tiempo.
                title('SEÑAL FM');
                xlabel('Tiempo (s)')
                ylabel('Magnitud (V)')
                grid on;
                axes(handles.espectro);
                plotspec(y_fm,Ts,4800,1);       %Dibuja la Señal en el dominio de la Frecuencia.
                n = floor(log(length(y_fm))/log(2));
                N = 2^n;
                ssf = (-N/2:N/2-1)/(Ts*N);
                set(handles.text11,'String',4800);
                set(handles.rango,'Value',1);
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
                        zoom off;
                        set(handles.zoom,'String','ZOOM OFF');
                end
                %*-*-*-*-*-*-

end

guidata(hObject,handles)


% --- Executes during object creation, after setting all properties.
function frec_c_CreateFcn(hObject, eventdata, handles)
% hObject    handle to frec_c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes during object creation, after setting all properties.
function v_f_m_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_f_m (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes on button press in zoom.
function zoom_Callback(hObject, eventdata, handles)
% hObject    handle to zoom (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a = get(hObject,'Value');
if a == 1
        zoom on;
        set(handles.zoom,'String','ZOOM ON');
else
        zoom off;
        set(handles.zoom,'String','ZOOM OFF');
end

% --- Executes during object creation, after setting all properties.
function uipanel1_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on slider movement.
function rango_Callback(hObject, eventdata, handles)
% hObject    handle to rango (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%       get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.ran = get(hObject,'Value');             %Carga en handles.slider1 el valor del Slider
handles.ran = 2000.*handles.ran;
if (handles.ran == 0)
            handles.ran = 1.5;
end
%set(handles.text11,'String',handles.ran);                  %Escribe el valor de Slider en static text

fun = get(handles.popupmenu1,'Value');

f = 1000*get(handles.frecuencia,'Value');
if (f == 0)
            f = 1;
end

fg = handles.ran;

handles.cic = 20 * get(handles.ciclos,'Value');
if (handles.cic == 0)
            handles.cic = 0.1;
end

handles.ejex1 = 0:1/(64*f):64/f;

Ts1 = 1/(64*f);

f_m = 1000 * get(handles.frec_m,'Value');
if (f_m == 0)
            f_m = 1;
end

f_c = 1000 * get(handles.frec_c,'Value');
if (f_c == 0)
            f_c = 1;
end

handles.ejex2 = 0:1/(64*f_c):64/f_c;
Ts = 1/(64*f_c);
Fs = 1/Ts;

switch fun
            case 1
                        y1 = sin(2*pi*f*handles.ejex1);
                        axes(handles.grafica);
                        dibujarGrafica(handles.ejex1,1,handles.cic,y1,f);          %Dibuja la Señal en el dominio del Tiempo.
                        title('SEÑAL GENERADA EN EL TIEMPO');
                        axes(handles.espectro);
                        plotspec(y1,Ts1,fg,fun);              %Dibuja la Señal en el dominio de la Frecuencia.
                        set(handles.text11,'String',fg);
```

```
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 2
        y2 = cos(2*pi*f*handles.ejex1);
        axes(handles.grafica);
        dibujarGrafica(handles.ejex1,1,handles.cic,y2,f);          %Dibuja la Señal en el dominio del Tiempo.
        title('SEÑAL GENERADA EN EL TIEMPO');
        axes(handles.espectro);
        plotspec(y2,Ts1,fg,fun);              %Dibuja la Señal en el dominio de la Frecuencia.
        set(handles.text11,'String',fg);

        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 3
        y3 = square(2*pi*f*handles.ejex1);
        axes(handles.grafica);
        dibujarGrafica(handles.ejex1,1.5,handles.cic,y3,f);          %Dibuja la Señal en el dominio del Tiempo.
        title('SEÑAL GENERADA EN EL TIEMPO');
        axes(handles.espectro);
        n = floor(log(length(y3))/log(2));
        N = 2^n;
        ssf = (-N/2:N/2-1)/(Ts1*N);
        plotspec(y3,Ts1,(round(max(ssf))*(fg/2000)),1);          %Dibuja la Señal en el dominio de la Frecuencia.
        set(handles.text11,'String',(round(max(ssf))*(fg/2000)));
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 4
        y4 = sawtooth(2*pi*f*handles.ejex1);
        axes(handles.grafica);
        dibujarGrafica(handles.ejex1,(1.5),handles.cic,y4,f);          %Dibuja la Señal en el dominio del Tiempo.
        title('SEÑAL GENERADA EN EL TIEMPO');
        axes(handles.espectro);
        n = floor(log(length(y4))/log(2));
        N = 2^n;
        ssf = (-N/2:N/2-1)/(Ts1*N);
```

```
plotspec(y4,Ts1,(round(max(ssf))*(fg/2000)),1);          %Dibuja la Señal en el dominio de la Frecuencia.
set(handles.text11,'String',(round(max(ssf))*(fg/2000)));
title('ESPECTRO');
a = get(handles.zoom,'Value');
if a == 1
        zoom on;
        set(handles.zoom,'String','ZOOM ON');
else
        zoom off;
        set(handles.zoom,'String','ZOOM OFF');
end
%*-*-*-*-*-*-

case 5
        y_m = cos(2*pi*f_m*handles.ejex2) + 1;
        axes(handles.grafica);
        %Dibuja la Señal en el dominio del Tiempo.
        plot(handles.ejex2,y_m,'--','LineWidth',1,'Color',[1 0 0]);          hold on;
        plot(handles.ejex2,-y_m,'--','LineWidth',1,'Color',[1 0 0]);         hold on;
        y_c = cos(2*pi*f_c*handles.ejex2);
        y_am = y_m.*y_c;
        plot(handles.ejex2,y_am,'LineWidth',2,'color',[0 1 0])
        hold off;
        set(gca,'Color',[0.16 0.16 0.16]);
        title('SEÑAL AM');
        xlabel('Tiempo (s)')
        ylabel('Magnitud (V)')
        grid on;
        axis([0 (handles.cic/f_m) -2.5 2.5]);
        axes(handles.espectro);
        plotspec(y_am,Ts,((2.5*f_c)*(fg/2000)),1);      %Dibuja la Señal en el dominio de la Frecuencia.
        set(handles.text11,'String',((2.5*f_c)*(fg/2000)));

        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 6
        x_m1 = cos(2*pi*f_m*handles.ejex2);
        y_fm = modulate(x_m1,f_c,Fs,'fm');
        axes(handles.grafica);
        %Dibuja la Señal en el dominio del Tiempo.
        dibujarGrafica(handles.ejex2,1.5,handles.cic,y_fm,f_m);          %Dibuja la Señal en el dominio del
Tiempo.
        title('SEÑAL FM');
        xlabel('Tiempo (s)')
        ylabel('Magnitud (V)')
        grid on;
        axes(handles.espectro);
        plotspec(y_fm,Ts,((4800)*(fg/2000)),1);      %Dibuja la Señal en el dominio de la Frecuencia.
        set(handles.text11,'String',((4800)*(fg/2000)));

        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
```

```
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

end

guidata(hObject,handles)
% --- Executes during object creation, after setting all properties.
function rango_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rango (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function ciclos_Callback(hObject, eventdata, handles)
% hObject    handle to ciclos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%       get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.cic = get(hObject,'Value');             %Carga en handles.slider1 el valor del Slider
handles.cic=20.*handles.cic;
if (handles.cic == 0)
        handles.cic = 0.1;
end
set(handles.text10,'String',handles.cic);       %Escribe el valor de Slider en static text

fun = get(handles.popupmenu1,'Value');

f = 1000*get(handles.frecuencia,'Value');
if (f == 0)
        f = 1;
end

fg = 2000*get(handles.rango,'Value');
%fg_1 = 6000 * get(handles.rango,'Value');

handles.ejex1 = 0:1/(64*f):64/f;

Ts1 = 1/(64*f);

f_m = 1000 * get(handles.frec_m,'Value');
if (f_m == 0)
        f_m = 1;
end
f_c = 1000 * get(handles.frec_c,'Value');
if (f_c == 0)
        f_c = 1;
end

handles.ejex2 = 0:1/(64*f_c):64/f_c;
Ts = 1/(64*f_c);
```

```
Fs = 1/Ts;

switch fun
        case 1
                y1 = sin(2*pi*f*handles.ejex1);
                axes(handles.grafica);
                dibujarGrafica(handles.ejex1,1,handles.cic,y1,f);       %Dibuja la Señal en el dominio del Tiempo.
                title('SEÑAL GENERADA EN EL TIEMPO');
                axes(handles.espectro);
                plotspec(y1,Ts1,(2*f),fun);       %Dibuja la Señal en el dominio de la Frecuencia.
                set(handles.text11,'String',(2*f));             %Escribe el valor de Slider en static text
                set(handles.rango,'Value',((2*f)/2000));
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
                        zoom off;
                        set(handles.zoom,'String','ZOOM OFF');
                end
                %*-*-*-*-*-*-

        case 2
                y2 = cos(2*pi*f*handles.ejex1);
                axes(handles.grafica);
                dibujarGrafica(handles.ejex1,1,handles.cic,y2,f);       %Dibuja la Señal en el dominio del Tiempo.
                title('SEÑAL GENERADA EN EL TIEMPO');
                axes(handles.espectro);
                plotspec(y2,Ts1,(2*f),fun);       %Dibuja la Señal en el dominio de la Frecuencia.
                set(handles.text11,'String',(2*f));             %Escribe el valor de Slider en static text
                set(handles.rango,'Value',((2*f)/2000));
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
                        zoom off;
                        set(handles.zoom,'String','ZOOM OFF');
                end
                %*-*-*-*-*-*-

        case 3
                y3 = square(2*pi*f*handles.ejex1);
                axes(handles.grafica);
                dibujarGrafica(handles.ejex1,1.5,handles.cic,y3,f);       %Dibuja la Señal en el dominio del Tiempo.
                title('SEÑAL GENERADA EN EL TIEMPO');
                axes(handles.espectro);
                plotspec(y3,Ts1,fg,fun);          %Dibuja la Señal en el dominio de la Frecuencia.
                n = floor(log(length(y3))/log(2));
                N = 2^n;
                ssf = (-N/2:N/2-1)/(Ts1*N);
                set(handles.text11,'String',round(max(ssf)));
                set(handles.rango,'Value',1);
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
                        zoom off;
```

```
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 4
        y4 = sawtooth(2*pi*f*handles.ejex1);
        axes(handles.grafica);
        dibujarGrafica(handles.ejex1,(1.5),handles.cic,y4,f);       %Dibuja la Señal en el dominio del Tiempo.
        title('SEÑAL GENERADA EN EL TIEMPO');
        axes(handles.espectro);
        plotspec(y4,Ts1,fg,fun);            %Dibuja la Señal en el dominio de la Frecuencia.
        n = floor(log(length(y4))/log(2));
        N = 2^n;
        ssf = (-N/2:N/2-1)/(Ts1*N);
        set(handles.text11,'String',round(max(ssf)));
        set(handles.rango,'Value',1);
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 5
        y_m = cos(2*pi*f_m*handles.ejex2) + 1;
        axes(handles.grafica);
        %Dibuja la Señal en el dominio del Tiempo.
        plot(handles.ejex2,y_m,'--','LineWidth',1,'Color',[1 0 0]);          hold on;
        plot(handles.ejex2,-y_m,'--','LineWidth',1,'Color',[1 0 0]);         hold on;
        y_c = cos(2*pi*f_c*handles.ejex2);
        y_am = y_m.*y_c;
        plot(handles.ejex2,y_am,'LineWidth',2,'color',[0 1 0])
        hold off;
        set(gca,'Color',[0.16 0.16 0.16]);
        title('SEÑAL AM');
        xlabel('Tiempo (s)')
        ylabel('Magnitud (V)')
        grid on;
        axis([0 (handles.cic/f_m) -2.5 2.5]);
        axes(handles.espectro);
        plotspec(y_am,Ts,2.5*f_c,fun);              %Dibuja la Señal en el dominio de la Frecuencia.
        set(handles.text11,'String',2.5*f_c);
        set(handles.rango,'Value',1);
        title('ESPECTRO');
        a = get(handles.zoom,'Value');
        if a == 1
                zoom on;
                set(handles.zoom,'String','ZOOM ON');
        else
                zoom off;
                set(handles.zoom,'String','ZOOM OFF');
        end
        %*-*-*-*-*-*-

case 6
        x_m1 = cos(2*pi*f_m*handles.ejex2);
        y_fm = modulate(x_m1,f_c,Fs,'fm');
        axes(handles.grafica);
```

```
                %Dibuja la Señal en el dominio del Tiempo.
                dibujarGrafica(handles.ejex2,1.5,handles.cic,y_fm,f_m);          %Dibuja la Señal en el dominio del
        Tiempo.
                title('SEÑAL FM');
                xlabel('Tiempo (s)')
                ylabel('Magnitud (V)')
                grid on;
                axes(handles.espectro);
                plotspec(y_fm,Ts,4800,1);       %Dibuja la Señal en el dominio de la Frecuencia.
                n = floor(log(length(y_fm))/log(2));
                N = 2^n;
                ssf = (-N/2:N/2-1)/(Ts*N);
                set(handles.text11,'String',4800);
                set(handles.rango,'Value',1);
                title('ESPECTRO');
                a = get(handles.zoom,'Value');
                if a == 1
                        zoom on;
                        set(handles.zoom,'String','ZOOM ON');
                else
                        zoom off;
                        set(handles.zoom,'String','ZOOM OFF');
                end
                %*-*-*-*-*-*-
end

guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.
function ciclos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ciclos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu3 contents as cell array
%       contents{get(hObject,'Value')} returns selected item from popupmenu3

% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end
```

ANEXO C

FUNCIONAMIENTO DEL MODO ZOOM EN LA TARJETA DE2

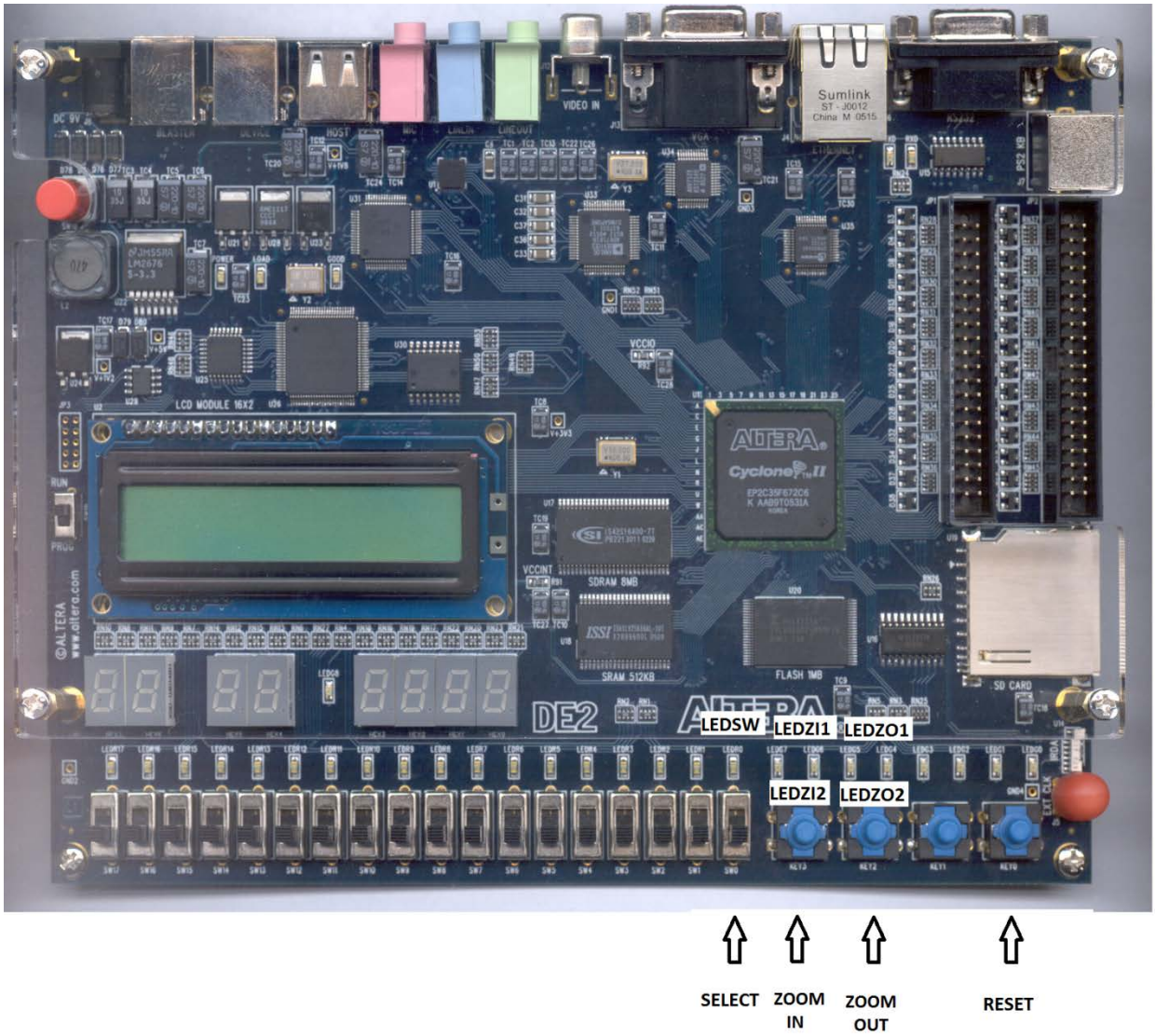


Figura 1 Función de los botones de la tarjeta DE2 en el proyecto

En la figura 1 observamos la imagen de la tarjeta DE2 haciendo referencia a los botones que se utilizan en la función de zoom de nuestro proyecto.

Solamente utilizamos 5 Leds, un switch y 3 botones, descritos a continuación:

- El switch ***SELECT*** que es usado para seleccionar la gráfica a ser amplificada o reducida.
- Los botones ***ZOOM IN*** y Z***OOM OUT*** permiten reducir y amplicar la imagen respectivamente, mientras que el botón ***RESET*** reinicia el sistema.
- Los cinco LEDs sirven de indicadores del estado de cada switch y botón usado. El ***LEDSW*** se enciende cuando se está apuntando al espectro de frecuencia de la señal, mientras que cuando se apunta a la gráfica del tiempo permanece apagado. ***LEDZI1*** y ***LEDZI2*** se encienden al presionar el botón de ***ZOOM IN***. Al presionar el botón ***ZOOM OUT***, los LEDs que se activan son ***LEDZO1*** y ***LEDZO2***.

Al arrancar nuestro proyecto en NIOS II, la tarjeta grafica en el monitor VGA la cuadrícula en espera a que los datos de la señal a graficar sean enviados, tal como se muestra en la figura 2. La flecha blanca mostrada en la imagen indica el gráfico sobre el cual se va a aplicar el zoom, en este caso se observa que el seleccionado es el correspondiente a la onda en el tiempo.

Figura 2 Gráfica en el tiempo seleccionada

Con el uso del switch ***SELECT*** (ver figura 1) podemos mover la flecha selectora de gráfica, es decir, si antes se había seleccionado la onda en el tiempo ahora se pasó a apuntar al gráfico correspondiente al espectro de frecuencias de la señal, como se observa en la figura 3.

Figura 3 Gráfica del espectro seleccionada

Para probar el funcionamiento de las opciones de zoom, se ha enviado desde MATLAB una señal modulada en frecuencia con frecuencia de mensaje 50 Hz y de portadora 500 Hz. Los resultados arrojados por la tarjeta se muestran en la figura 4, y la flecha selectora está sobre la gráfica de la señal en el tiempo, lo que implica que toda variación de zoom se hace únicamente sobre esta. Tal como está en la figura 4, la señal en el tiempo tiene ZOOM x1.

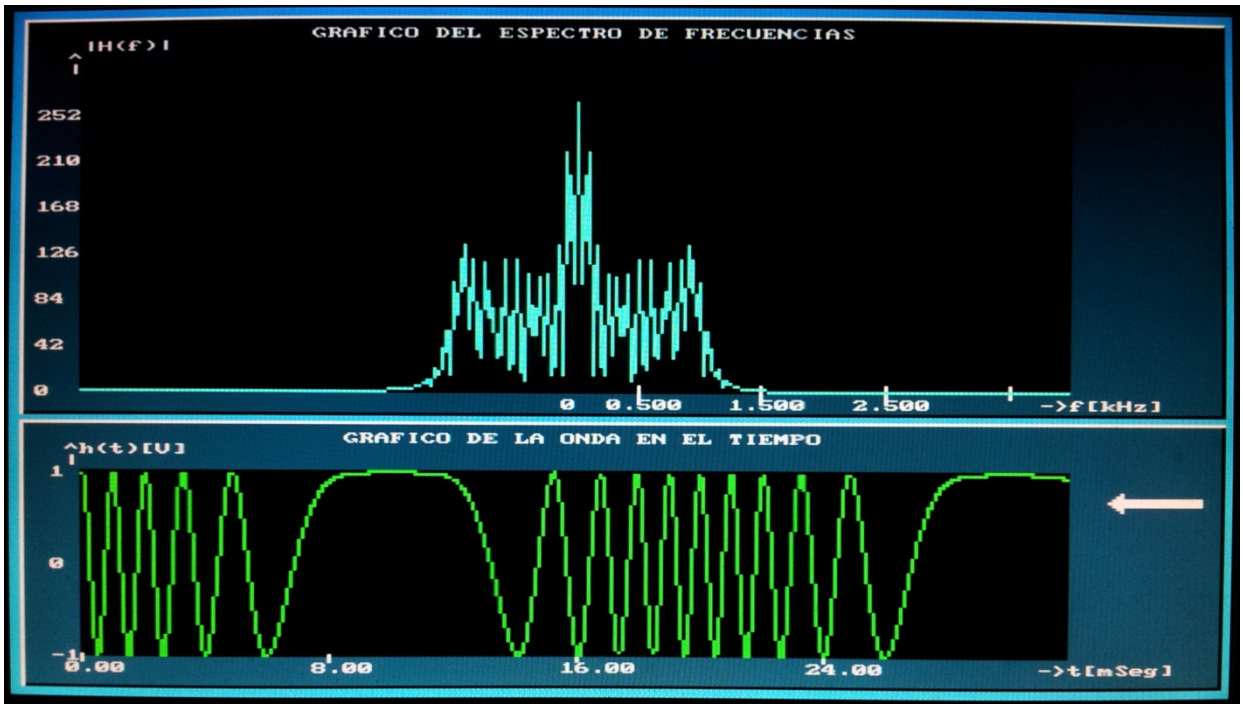


Figura 4 Gráfica de la señal FM en el dominio del tiempo ZOOM IN x1

Para hacer un zoom in en la gráfica debemos presionar el botón ***ZOOM IN*** (ver figura 1) y el resultado es el mostrado en la figura 5.

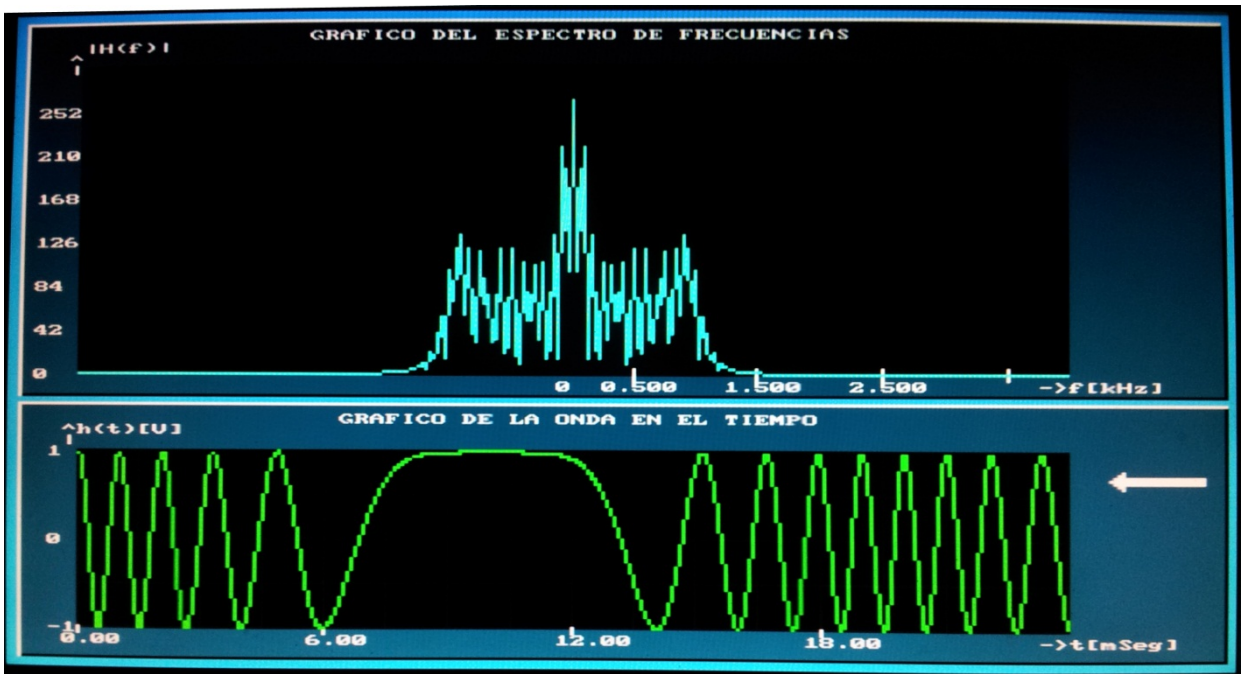


Figura 5 Gráfica de la señal FM en el dominio del tiempo ZOOM IN x2

En la figura 5 tenemos como resultado que se realizó una ampliación de la señal en el dominio del tiempo en comparación con la mostrada en la figura 4. A este nivel de ampliación lo hemos denominado ZOOM IN x2.

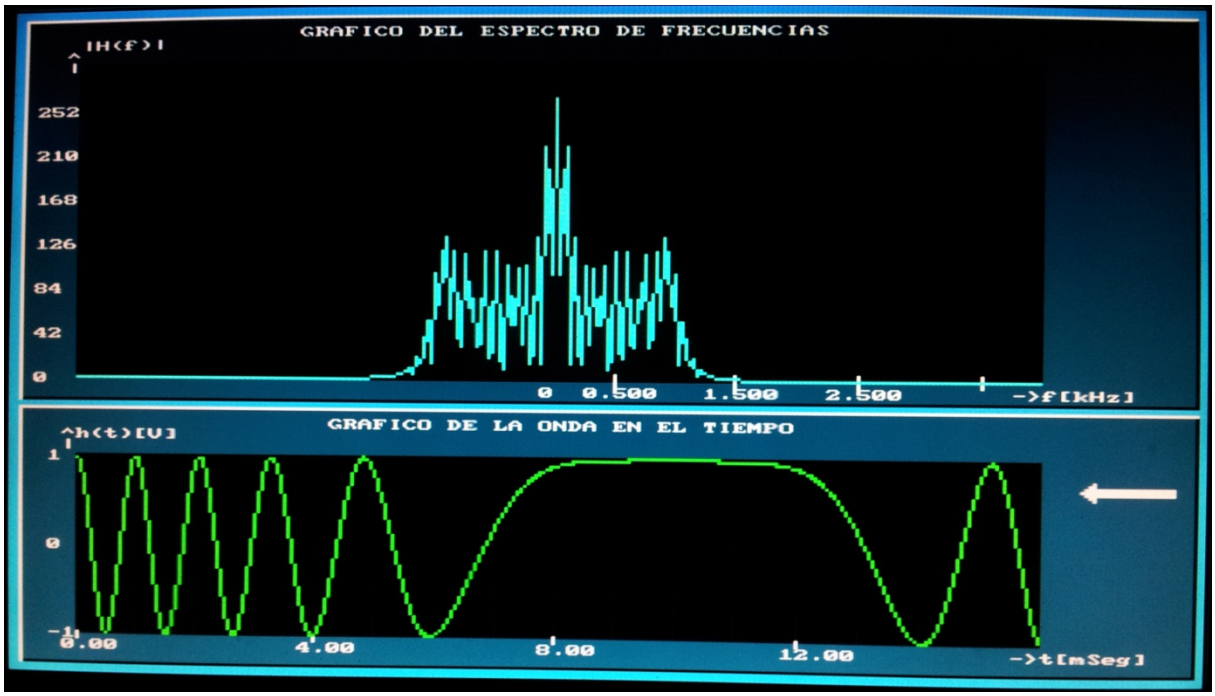


Figura 6 Gráfica de la señal FM en el dominio del tiempo ZOOM IN x3

Si continuamos presionando el botón ***ZOOM IN*** conseguiremos niveles de ampliación mayores, a los que hemos denominado ZOOM IN x3 y ZOOM IN x4 y se muestran en las figura 6 y 7 respectivamente, siendo ZOOM IN x4 el máximo valor de ampliación soportado por nuestro proyecto.

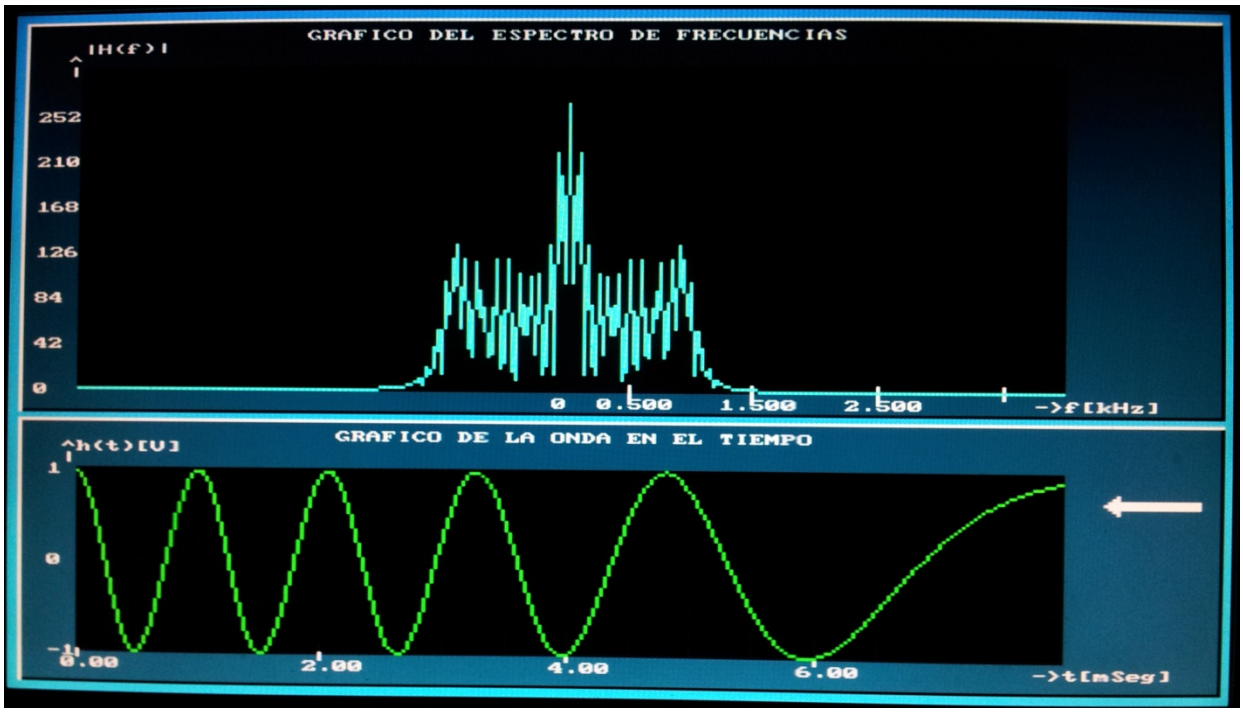


Figura 7 Gráfica de la señal FM en el dominio del tiempo ZOOM IN x3

Si continuamos presionando el botón ***ZOOM IN*** conseguiremos niveles de ampliación mayores, a los que hemos denominado ZOOM IN x3 y ZOOM IN x4 y se muestran en las figura 6 y 7 respectivamente, siendo ZOOM IN x4 el máximo valor de ampliación soportado por nuestro proyecto.

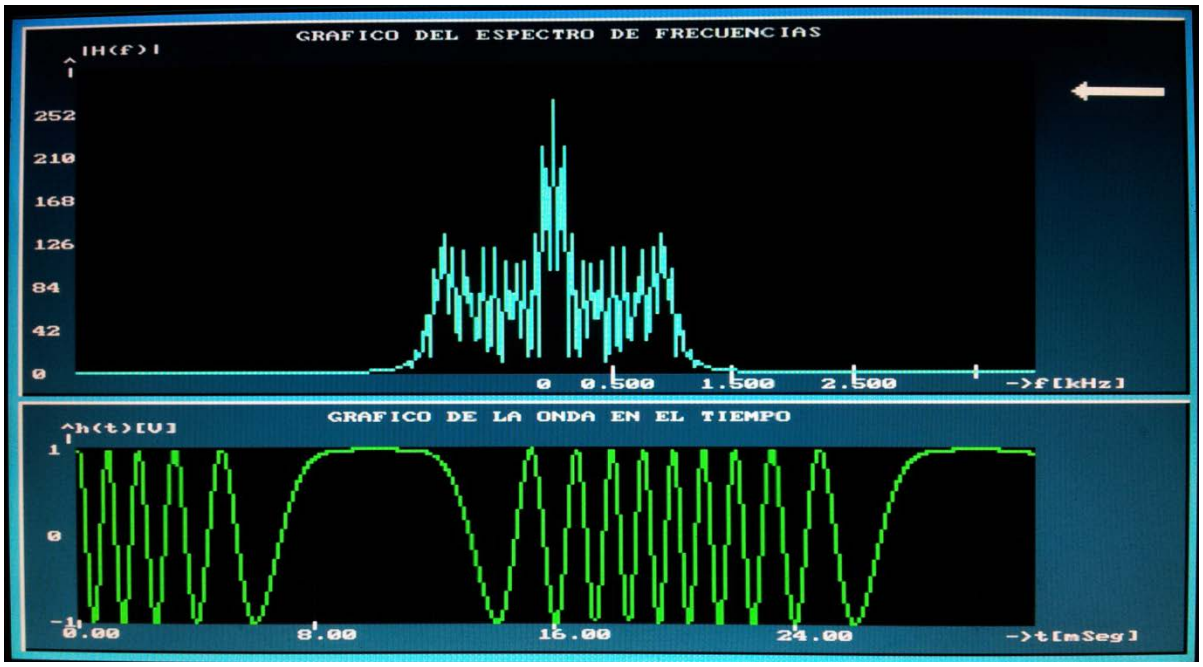


Figura 8 Gráfica de la señal FM en el dominio de la frecuencia ZOOM IN x1

Al cambiar de estado al switch ***SELECT***, la flecha apunta al gráfico de espectro en frecuencia, el mismo que queda seleccionado para realizar cambios en el zoom como se muestra en la figura 8. En este espectro podemos ver la utilidad del zoom en nuestro proyecto, ya que no es muy apreciable, por lo que resulta necesario hacer una ampliación para obtener mayor detalle en el gráfico.

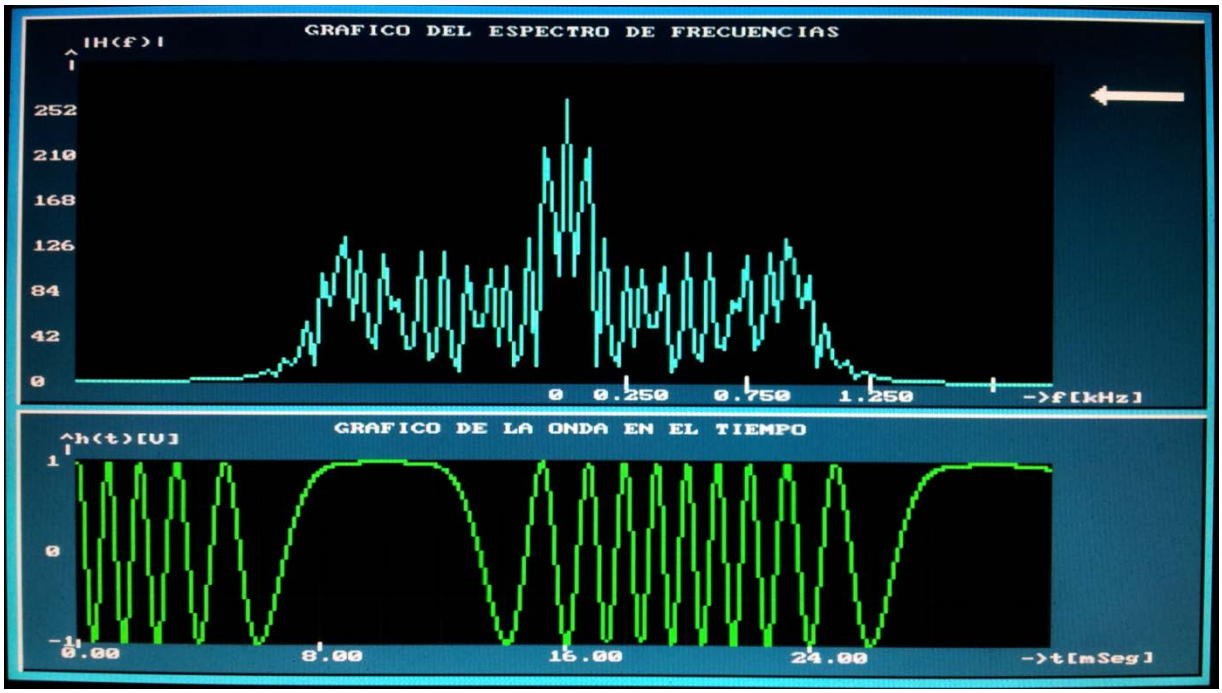


Figura 9 Gráfica de la señal FM en el dominio de la frecuencia ZOOM IN x2

Al igual que con el gráfico de la señal FM en el dominio del tiempo, al presionar el botón ***ZOOM IN*** logramos ampliar la imagen del espectro de frecuencia, tal como lo muestra la figura 9. Este aumento lo llamamos ZOOM IN x2.

Si seguimos presionando el botón ***ZOOM IN*** obtendremos los resultados mostrados en las figuras 10, nivel de aumento denominado ZOOM IN x3.

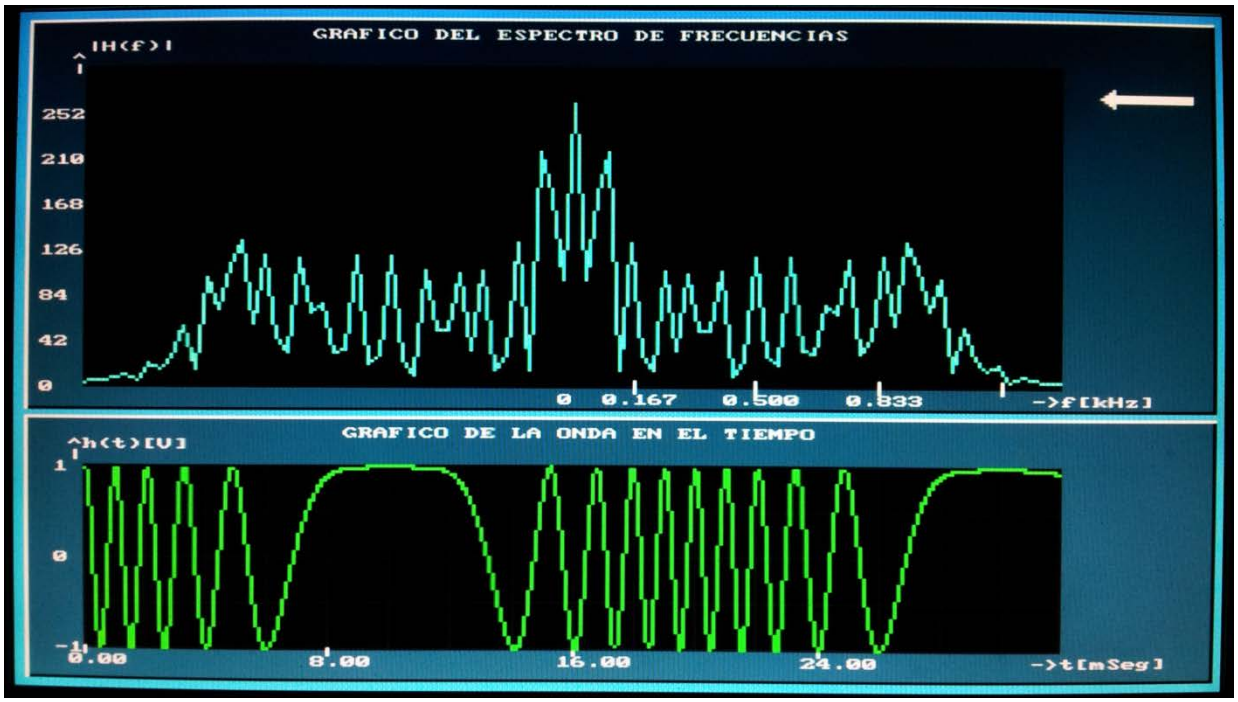


Figura 10 Gráfica de la señal FM en el dominio de la frecuencia ZOOM IN x3

Cabe mencionar que para el caso de ZOOM IN en el gráfico del espectro de frecuencia, se pueden obtener niveles de ampliación muy altos como el mostrado en la figura 11, correspondiente a ZOOM IN x10.

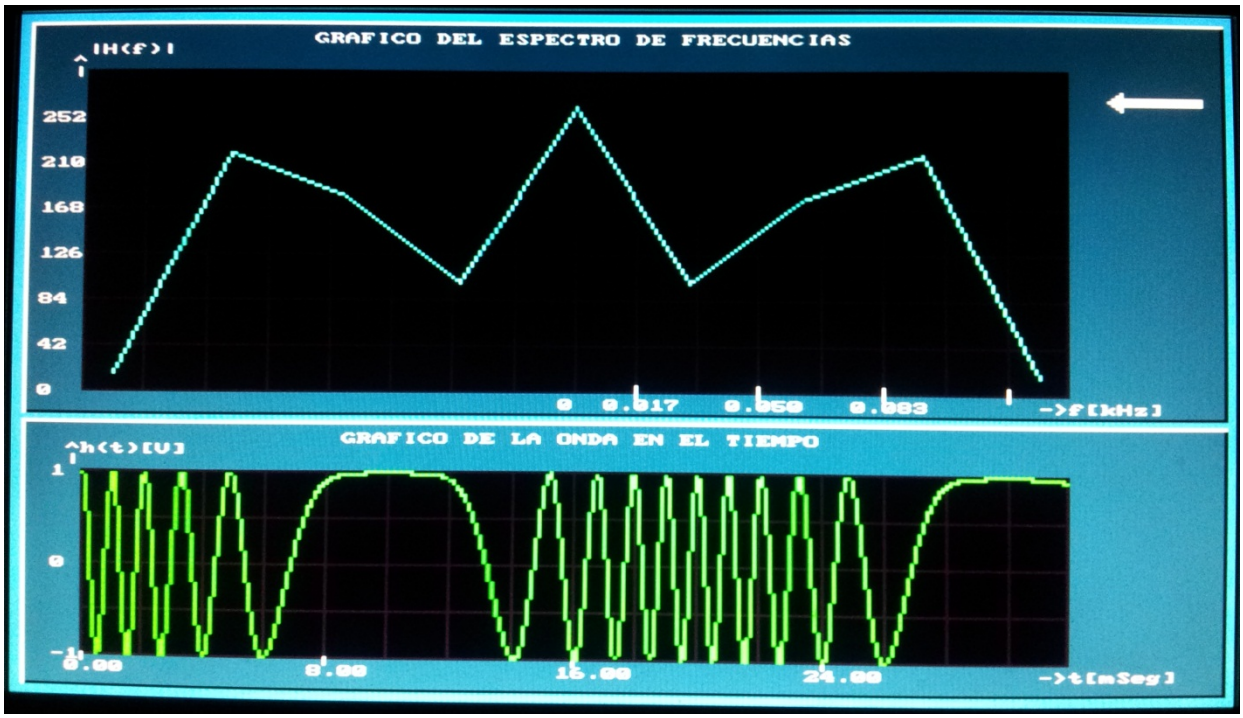


Figura 11 Gráfica de la señal FM en el dominio de la frecuencia ZOOM IN x10

Si a partir del gráfico mostrado en la figura 11 se presiona el botón ***ZOOM OUT***, lo que se obtendría como resultado es un ZOOM IN x9, y si vuelve a presionar ***ZOOM OUT*** tendríamos un ZOOM IN x8 y asi hasta llegar a la imagen original mostrada en la figura 8 correspondiente a ZOOM IN x1.

Para una explicación de la opción zoom out hemos enviado una señal Diente de Sierra de frecuencia de 200 Hz mostrada en la figura 12.

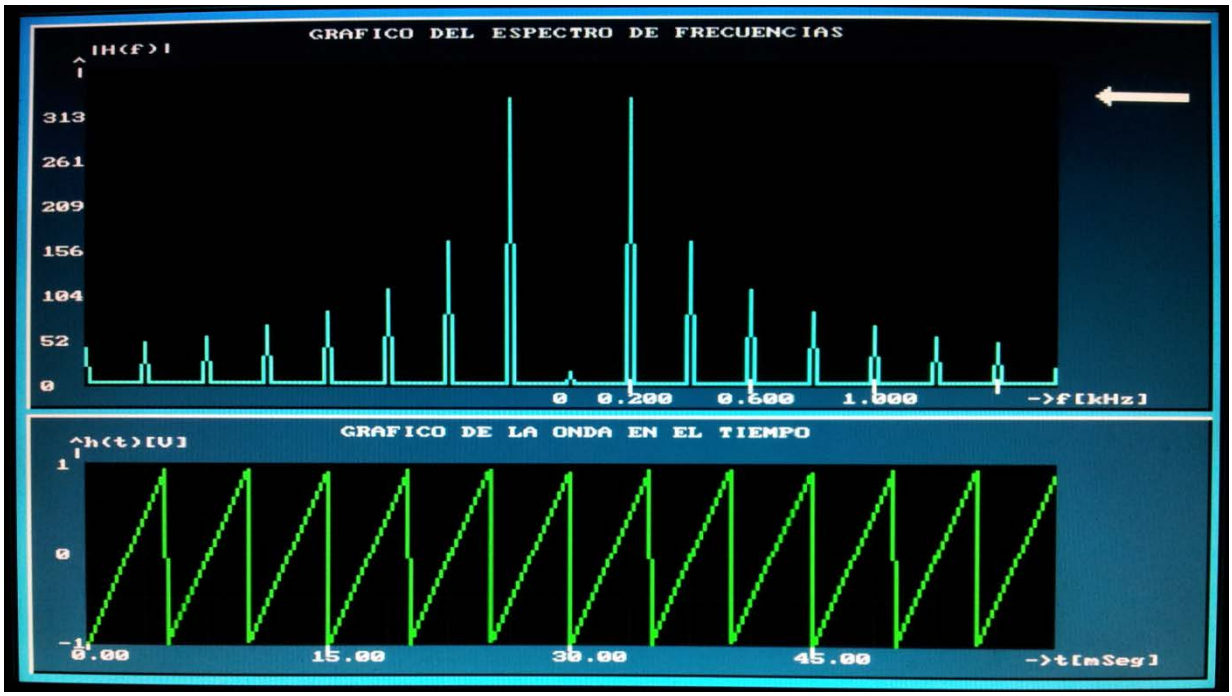


Figura 12 Gráfica de la señal Diente de Sierra en el dominio de la frecuencia ZOOM OUT x1

Al presionar el botón ***ZOOM OUT*** estaríamos reduciendo al gráfico correspondiente al espectro de frecuencia, como se muestra en la figura 13 que corresponde a un ZOOM OUT x2. Esta opción de zoom es de mucha ayuda en caso de que el usuario de este proyecto quisiera ver la gráfica para un mayor rango de frecuencias.

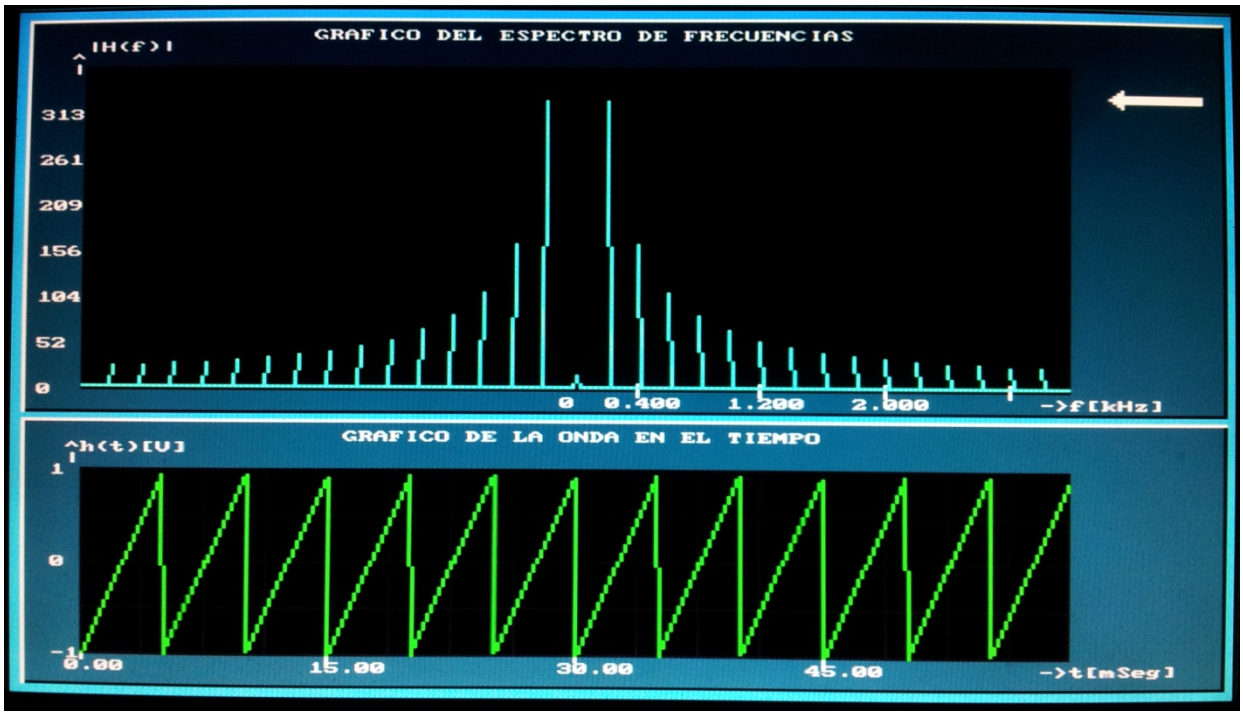


Figura 13 Gráfica de la señal Diente de Sierra en el dominio de la frecuencia ZOOM OUT x2

Para aumentar el rango de frecuencias a graficar en el espectro nuevamente se hace uso del botón ***ZOOM OUT*** obteniéndose asi la reducción correspondiente a un ZOOM OUT x3, tal como se muestra en la figura 14.

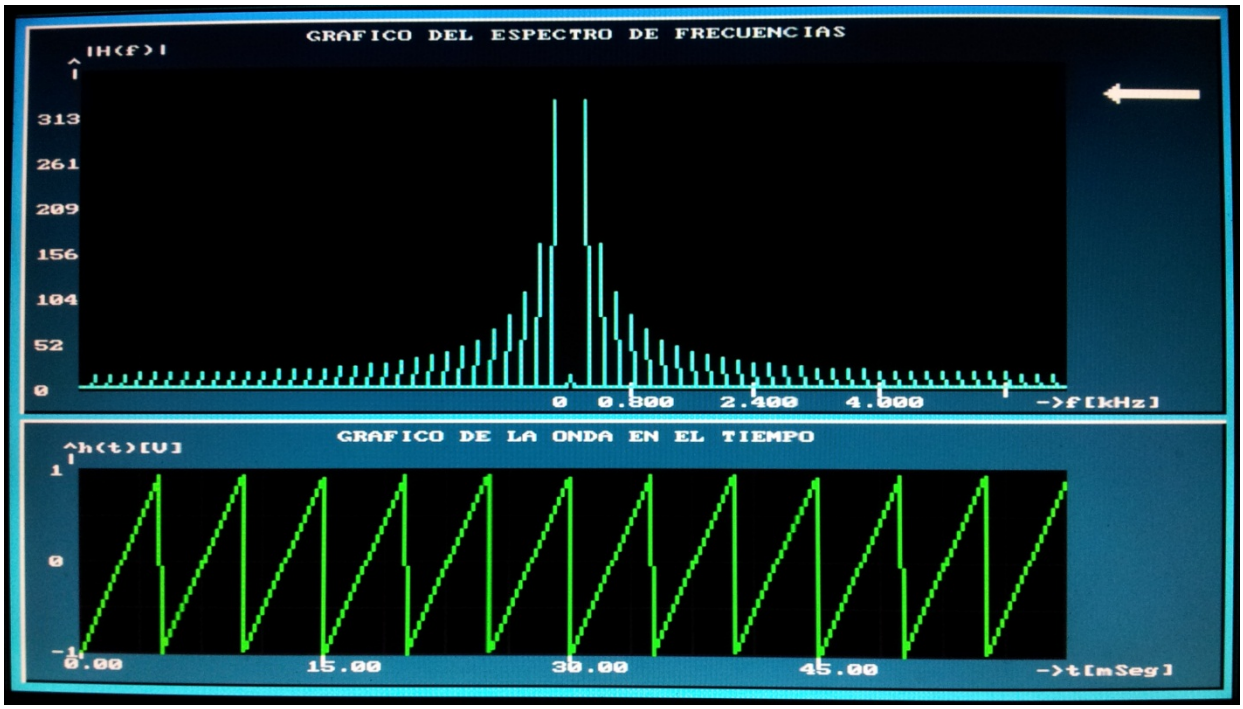


Figura 14 Gráfica de la señal Diente de Sierra en el dominio de la frecuencia ZOOM OUT x3