DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PORTÁTIL DE REGISTRO DE SEÑALES MICRO-SÍSMICAS

BRENDA NATALIA ROSERO LONDOÑO
MARÍA JOSÉ CADENA VINUEZA

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
SAN JUAN DE PASTO
2009

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PORTÁTIL DE REGISTRO DE SEÑALES MICRO-SÍSMICAS

BRENDA NATALIA ROSERO LONDOÑO
MARÍA JOSÉ CADENA VINUEZA

PRESENTADO COMO REQUISITO PARCIAL PARA OPTAR AL TÍTULO DE INGENÍERO ELECTRÓNICO

DIRECTOR:
ING. DARIO FERNANDO FAJARDO FAJARDO

CODIRECTOR:
PHD. HUGO CORAL MONCAYO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
SAN JUAN DE PASTO
2009

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

_____

_____

Firma del jurado

_____

Firma del jurado

San Juan de Pasto, noviembre de 2009

"Las ideas y conclusiones aportadas en la tesis de grado son responsabilidad exclusiva de los autores"
Artículo 1. del acuerdo No. 324 del 11 de Octubre de 1966, emanado por el Honorable Consejo Directivo de la Universidad de Nariño.

A Aquel en cuya mano está el alma de todo viviente,
Y el hálito de todo el género humano.

Al único y sabio Dios, sea gloria
mediante Jesucristo para siempre.
Amén.

AGRADECIMIENTOS

Mg. Dario Fernando Fajardo Fajardo director del trabajo de grado, por su apoyo, tiempo y dirección para el desarrollo y culminación del mismo.

Phd. Hugo Coral Moncayo co-director del trabajo de grado y director del Centro de Investigación en Ingeniería Sismológica de la Universidad de Nariño, por su generosidad al disponer del rubro asignado al Centro de Investigación para la financiación de gran parte de este trabajo; por su colaboración en pro del éxito de este proyecto.

Instructores del Sena Ing. Iván Vinueza y Gerardo Gamboa quienes nos guiaron en el diseño y construcción externa del equipo.

Programa de Ingeniería Electrónica de la Universidad de Nariño, docentes por permitir nuestra formación como profesionales; al personal de laboratorio por el préstamo de equipos.

Ing. Andrés Hillón Sarmiento por su disponibilidad y colaboración.

A nuestros familiares y amigos por su apoyo incondicional.

CONTENIDO

LISTA DE CUADROS

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

ACELERÓGRAFO: instrumento que registra la aceleración del terreno en función del tiempo.

ACELEROGRAMA: registro de la aceleración del terreno en un sitio dado en función del tiempo. La aceleración se registra generalmente en tres direcciones: dos componentes horizontales, ortogonales entre sí, y una vertical.

ACELERÓMETRO: sensor que mide la aceleración lineal a lo largo de su eje sensible. Convierte la aceleración de gravedad o de movimiento, en una señal eléctrica analógica proporcional a la fuerza aplicada al sistema; o mecanismo sometido a vibración o aceleración.

ADQUISICIÓN: recolección de un conjunto de variables físicas, conversión en voltaje y digitalización de manera que se puedan procesar en un ordenador.

ARRAY: variable que posee varias posiciones para almacenar un valor en cada posición. Las posiciones son accedidas mediante un índice numérico.

BIT DE RESOLUCIÓN: número de bits que el convertidor analógico a digital (ADC) utiliza para representar una señal.

CLASE: se puede considerar como un patrón para construir objetos. En C++, un objeto es sólo un tipo de variable de una clase determinada. Es importante distinguir entre objetos y clases, la clase es simplemente una declaración, no tiene asociado ningún objeto, de modo que no puede recibir mensajes ni procesarlos, esto únicamente lo hacen los objetos.

DATO: representación simbólica (numérica, alfabética...), atributo o característica de un valor. No tiene sentido en sí mismo, pero convenientemente tratado (procesado) se puede utilizar en la relación de cálculos o toma de decisiones.

DOPAJE: proceso de agregar impurezas en un semiconductor extremadamente puro, con el fin de cambiar sus propiedades eléctricas. Las impurezas utilizadas dependen del tipo de semiconductores a dopar.

DRIVER SOFTWARE: driver software normalmente viene con el hardware DAQ o de otros proveedores, y permite que el sistema operativo pueda reconocer el hardware DAQ y dar así a los programas acceso a las señales de lectura por el hardware DAQ. Un buen conductor ofrece un alto y bajo nivel de acceso.

ENTORNO DE DESARROLLO INTEGRADO (IDE): es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o puede utilizarse para varios.

FRECUENCIA DE ONDA: número de veces que se repite un proceso cíclico por unidad de tiempo; número de ciclos por unidad de tiempo de un proceso oscilatorio. La unidad de frecuencia es el hertz y se mide en ciclos por segundo; la frecuencia es el inverso del período de vibración o del período de onda.

INTERFAZ DE PROGRAMACIÓN DE APLICACIONES (API): serie de servicios o funciones que el sistema operativo ofrece al programador, como por ejemplo, imprimir un caracter en pantalla, leer el teclado y escribir en un fichero de disco. Visto desde la perspectiva del código máquina, el API aparece como una serie de llamadas, mientras que desde la de un lenguaje de alto nivel, el API aparece como un conjunto de procedimientos y funciones.

LENGUAJE DE PROGRAMACIÓN: es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.

MICROZONIFICACIÓN SÍSMICA: proceso de determinación de la amenaza sísmica en varios sitios con el propósito de delimitar zonas sujetas a un grado similar de riesgo.

NULL: el término null o nulo es a menudo utilizado en la computación, haciendo referencia a la nada.

OBJETO: unidad que engloba en sí mismo datos y procedimientos necesarios para el tratamiento de esos datos. Hasta ahora se habían hecho programas en los que los datos y las funciones estaban perfectamente separadas, cuando se programa con objetos esto no es así, cada objeto contiene datos y funciones. Y un programa se construye como un conjunto de objetos, o incluso como un único objeto.

ONDAS LOVE: son ondas superficiales que producen un movimiento horizontal de

corte en superficie. La velocidad de las ondas Love es ligeramente superior a la velocidad de las ondas Rayleigh.

ONDAS RAYLEIGH: son ondas superficiales que viajan a lo largo de la superficie de la tierra, tienen su máxima amplitud en la superficie libre, decrecen exponencialmente con la profundidad y su trayectoria de propagación es elíptica.

ONDAS SÍSMICAS: ondas elásticas generadas por un sismo que se propagan a partir del foco en todas direcciones. Vibración de la roca o partículas de terreno causada por un sismo.

ONDAS SUPERFICIALES: ondas que se propagan por la superficie de discontinuidad de la interfase de la superficie terrestre (tierra-aire y tierra-agua). Son las causantes de los daños producidos por los sismos en las construcciones.

PERÍODO DE ONDA: intervalo de tiempo entre dos crestas sucesivas en una onda sinusoidal; intervalo entre amplitudes máximas de ondas sísmicas. El período se mide en segundos y es el inverso de la frecuencia de onda.

RANGO: valores máximo y mínimo entre los que el sensor, instrumento o dispositivo, funcionan bajo unas especificaciones.

SISMO: evento físico causado por la liberación repentina de energía debido a una dislocación o desplazamiento en la corteza terrestre. Parte de la energía es irradiada en todas las direcciones en forma de ondas elásticas u ondas sísmicas. Es percibido en la superficie como una vibración del terreno; se le denomina temblor cuando no causa daños y terremoto cuando la sacudida es violenta y el evento es destructivo, causando daños severos o víctimas.

SISMOLOGÍA: rama de la ciencia que estudia los sismos, las fuentes sísmicas y la propagación de las ondas sísmicas a través del medio sólido o líquido de la Tierra.

SISTEMA: conjunto organizado de dispositivos que interactúan entre sí ofreciendo prestaciones más completas y de más alto nivel. Una vez que las señales eléctricas se transforman en digitales, se envían a través del bus de datos a la memoria del PC. Una vez los datos están en la memoria pueden procesarse con una aplicación adecuada.

RESUMEN

En el campo de la instrumentación sísmica, específicamente en aquellos sistemas o equipos encargados de la medición de microsismos, la realización de pruebas de campo es de mucha importancia, puesto que éstas proporcionan los datos necesarios para establecer el comportamiento de los suelos en distintas zonas de una determinada ciudad o región.

Por esta razón, se implementa un sistema portátil de registro de señales microsísmicas que consta de un sensor o acelerómetro, una etapa de acondicionamiento y una etapa de adquisición de datos. En el acondicionamiento de la señal se realiza una implementación física de amplificadores y filtros que adecúan la señal para su adquisición.

Se desarrolla un software de interfaz con el usuario que permite controlar los parámetros de adquisición y registro y además efectuar un procesamiento de los datos.

Utilizando señales conocidas se verifica el correcto funcionamiento del equipo y se realizan correcciones finales mediante pruebas de campo simultáneas con el acelerógrafo CUSP-3C de la empresa Canterbury Seismic, perteneciente al Centro de Investigación en Ingeniería Sismológica de la Universidad de Nariño.

Se garantiza la portabilidad del equipo incluyendo además de la alimentación por medio de la red eléctrica, una batería recargable.

ABSTRACT

In seismic instrumentation field, specifically the microseismic measurement systems, open field measures play an important role as they provide the data needed to establish the soils behavior in different parts of a particular city or region.

For this reason, a portable system is developed for recording microseismic signals, which consists in a sensor or accelerometer, a conditioning and an acquisition module. In the signal conditioning a physic implementation of amplifiers and filters is realized to adecuate the signal for its acquisition.

A software with a user interface is developed to control the acquisition and recording parameters and to perform a processing of the data.

In order to verify a well performance, several tests have been applied using known signals. In addition, for the final corrections simultaneous open field tests took place; they were made with the CUSP 3C accelerograph of Canterbury Seismic, property of the Sismological Engineering Research Center at the University of Nariño.

To ensure the portability, the system includes a rechargeable battery as well as the electric supply.

INTRODUCCIÓN

La medición de microsismos es una herramienta muy utilizada en la ingeniería sísmica para realizar estudios de amenaza sísmica y la microzonificación de una región. Como dice Hays[1], la microzonificación consiste en establecer zonas de suelos con comportamiento similar durante un sismo, de manera que puedan definirse allí recomendaciones precisas para el diseño y construcción de edificaciones sismo-resistentes.

Uno de los métodos utilizados para realizar mapas de microzonificación sísmica, es la elaboración de pruebas de campo en diversas zonas de la ciudad utilizando un equipo capaz de detectar microsismos, que permita hacer un registro de ellos para su posterior análisis. En este sentido, este proyecto tiene como objetivo principal, el diseño y la implementación de un sistema portátil de adquisición, registro y procesamiento de señales microsísmicas que pueda ser utilizado para realizar mediciones en diferentes lugares de la ciudad de San Juan de Pasto y en el departamento de Nariño.

El equipo consta de un instrumento de medida de aceleración o acelerómetro, una etapa de acondicionamiento de las señales obtenidas basada en amplificadores de instrumentación, una etapa de adquisición que consta de una tarjeta de adquisición de datos y una interfaz de software, destinada al registro y procesamiento de los datos.

En este trabajo se describe el diseño y la construcción del equipo así como el desarrollo del software de interfaz de usuario.

[1]HAYS, Walter. Aspectos fundamentales de la geología y la sismología para la microzonificación sísmica. En : Física de la Tierra. Vol. 1 (1989); p. 217-250.

1 INSTRUMENTACIÓN SÍSMICA

1.1. DEFINICIÓN Y CLASIFICACIÓN DE LOS SISMOS

El Instituto Colombiano de Geología y Minería INGEOMINAS define: "Los sismos corresponden al proceso de generación de ondas y su posterior propagación por el interior de la Tierra. Al llegar a la superficie, estas ondas se dejan sentir tanto por la población, como por la estructura y dependiendo de la amplitud del movimiento (desplazamiento, velocidad y aceleración del suelo) y de su duración, el sismo producirá mayor o menor intensidad"[1].

Las señales sísmicas pueden ser de una amplia gama de frecuencias, es por eso que para elegir los equipos en ingeniería sísmica se requiere un conocimiento del espectro de frecuencias de las vibraciones, que comprende desde el de las mareas terrestres con períodos de días, hasta los bajos períodos presentes en los modelos a escala reducida pasando, entre otras, por las frecuencias características de los sismos locales, las explosiones y las vibraciones industriales.

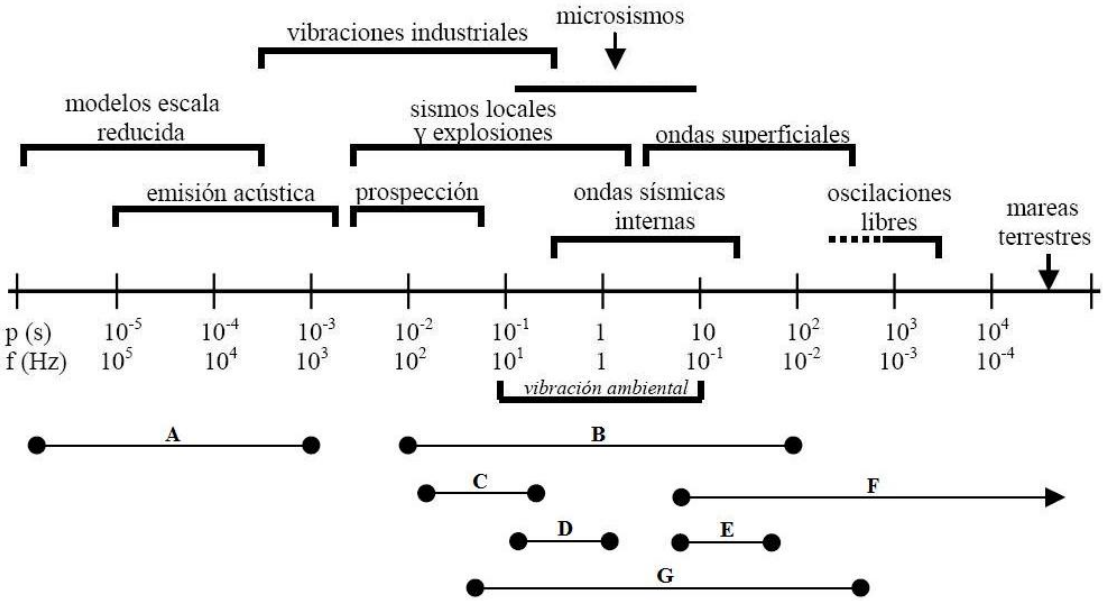
En la figura 1.1 se representa esquemáticamente el espectro sísmico en función del período y la frecuencia, señalando las fuentes y los instrumentos de medición en los intervalos que se encuentran especificados en el cuadro 1.1.

Cuadro 1.1. Relación entre el rango espectral y los diferentes instrumentos de medición sísmica

	Instrumento	**Rango espectral**
A	Piezoeléctricos	10^3 a 10^6 Hz
B	Acelerómetros	0.01 a 100 Hz
C	Geófonos de prospección	4 a 1500 Hz
D	Sismómetros de corto periodo	0.2 a 2 s
E	Sismómetros de largo periodo	10 a 100 s
F	Extensómetros, deformímetros, distanciómetros	DC a 0.5 Hz
G	Banda ancha	0.003 a 30 Hz

[1]Glosario Geológico Minero. [en línea]. Versión 1.8. Bogotá. Instituto Colombiano de Geología y minería, 2005. Disponible en Internet: `http://www.ingeominas.gov.co/index.php?option=com_glossary&catid=82&func=display&search=sismo`

Figura 1.1. Espectro sísmico



GÓMEZ, Marisol; ZORRILLA, Juan Carlos; MONSALVE, Jaramillo. La electrónica como soporte instrumental de la sismología. Observatorio Sismológico de la Universidad del Quindío. (2009) p. 2.

En general, los sismos se clasifican en: microsismos cuando son imperceptibles, macrosismos cuando son notados por el hombre y causan daños en enseres y casas y megasismos cuando son tan violentos que pueden producir la destrucción de edificios, ruina de ciudades y gran número de víctimas.

1.2. MICROTREPIDACIONES

Las microtrepidaciones son vibraciones del suelo de baja amplitud, conformadas por ondas sísmicas internas y superficiales que pueden ser de largo y corto período.

Las de largo período (mayores a un segundo), están conformadas por ondas superficiales Rayleigh, que se mueven en la dirección de propagación y ondas Love que provocan cortes horizontales en la tierra y son generadas por fuentes naturales, como: el viento, olas de mar y variaciones de presión de aire; las segundas, de período menor a un segundo, son producidas por fuentes artificiales tales como: el tráfico vehicular, actividad de plantas industriales y maquinarias.

La amplitud de las microtrepidaciones es muy dependiente de las condiciones locales del suelo, por lo tanto es difícil estimar rangos exactos para delimitar la amplitud de estos movimientos; sin embargo, a partir de investigaciones realizadas como las de Torres[2], se delimita el nivel de las vibraciones entre $\pm 10 \cdot 10^{-3} cm/s$. El rango de frecuencias de estos movimientos se extiende desde los 0,1 Hz hasta los 100 Hz, como se observa en el diagrama de la figura 1.1, este rango de frecuencias incluye desde las vibraciones ambientales o microsismos hasta algunas vibraciones industriales.

Como resultado de las mediciones de las microtrepidaciones se pueden conocer: las características de vibración de varias capas del suelo, la frecuencia fundamental de vibración, el desplazamiento y la velocidad del mismo; por lo tanto, se consideran una herramienta útil para adelantar estudios de dinámica de suelos, al permitir conocer la respuesta o comportamiento de éstos frente a las ondas sísmicas. Además, permiten adaptar los modelos de comportamiento del suelo a bajas deformaciones para luego ser usados en modelos de grandes deformaciones.

Como lo expresa María Luisa Bermudez y otros[3], para el análisis de microtrepidaciones se han propuesto tres técnicas que son: Interpretación Directa de la Transformada de Fourier, también llamada Amplitudes Espectrales, cálculo de los espectros relativos de amplificación de las estaciones en suelo blando versus una estación en suelo firme o Técnica de Kagami y cálculo de las relaciones espectrales entre la componente horizontal y vertical del movimiento en un mismo sitio o Técnica de Nakamura.

1.2.1. Microzonificación sísmica. Es la división de una región en zonas geográficas que se prevé, experimentarán la misma gravedad relativa de un peligro sísmico y es una parte importante del proceso de evaluación de la peligrosidad sísmica, es decir, la estimación de los fenómenos primarios y secundarios que acompañan a un sismo.

Las técnicas para realizar la microzonificación sísmica utilizan la medición de microtrepidaciones, una de ellas es la de Nakamura, la cual ha tenido bastante aceptación debido a su fácil implementación, tanto en el trabajo de campo como en el procesamiento de

[2]TORRES M., Gilbert Francisco. Importancia de la microzonificación sísmica de las principales ciudades del estado de Veracruz [en línea]. Disponible en Internet: `http://www.smsp.org.mx/rhigiene/docs/Importancia%20de%20la%20Microzonificaci%C3%B3n%20%28Torres%20Morales.%29.doc`

[3]BERMUDEZ, María Luisa et al. Uso de las microtrepidaciones para la evaluación de la respuesta dinámica de los suelos. [en línea], 2002. Disponible en internet: `http://bdrsnc.ingeominas.gov.co/publicaRNAC/PUBLICACIONES/SISMOLOGIA_2002/\MICROTREPIDACION_RESPUESTA_DINAMICA_DE_SUELOS.PDF`

los datos.

Alfaro, Navarro y Sánches[4] afirman que los mapas de microzonificación pueden utilizarse para planificar la construcción y el desarrollo urbano, para guiar el diseño resistente a los terremotos de los edificios nuevos, y reforzar los edificios e instalaciones existentes, además proporcionan beneficios como:

- La creación de un modelo regional mejorado de peligrosidad sísmica, que permitirá la identificación de las zonas de fallas.
- El establecimiento del nivel máximo de aceleración del suelo que se espera ocurra durante un tiempo determinado.
- La especificación de recomendaciones para mejorar los códigos de edificación y las prácticas de construcción y uso de la tierra.

1.3. INSTRUMENTOS DE REGISTRO SÍSMICO

Antoni Roca[5] precisa que, para conocer las características de las ondas sísmicas es necesario registrarlas de tal forma que puedan ser estudiadas posteriormente y determinar así la duración del movimiento, sus direcciones principales, etc.; para ello se emplean los instrumentos de registro sísmico.

Un instrumento de registro sísmico es un dispositivo capaz de proporcionar información sobre algún parámetro característico del movimiento del suelo. Normalmente, en cada instante de tiempo t, este instrumento proporcionará una salida $s(t)$, al ser sometido a una excitación de entrada $e(t)$, como se observa en la figura 1.2.

La entrada $e(t)$ es la vibración del punto de observación en cada instante de tiempo t, la amplitud de la vibración puede caracterizarse por el desplazamiento $z(t)$, la velocidad $\dot{z}(t) = \frac{dz}{dt}$ o la aceleración $\ddot{z}(t) = \frac{d^2z}{dt^2}$.

La salida, $s(t)$, es el registro obtenido, la cual puede obtenerse en general, por aplicación

[4]ALFARO, A.; NAVARRO, J.; SÁNCHES, J. Microzonificación sísmica de Barcelona utilizando el método de Nakamura Ventajas y Limitaciones. Barcelona: Universidad Politécnica de Cataluña. Departamento de Ingeniería del Terreno y Cartográfica. (1999) p. 1.

[5]ROCA, Antoni. Instrumentación para campo cercano y análisis de acelerogramas. En : Física de la Tierra. Vol. 1 (1989) p. 133-134.

Figura 1.2. Diagrama básico de un instrumento de registro sísmico

Fuente: ROCA, Antoni. Instrumentación para campo cercano y análisis de acelerogramas. En : Física de la Tierra. Vol. 1 (1989) p. 133.

de un operador H a la entrada $e(t)$, independientemente de la amplitud de la señal de entrada, como se expresa en la ecuación 1.1.

$$s(t) = H(e(t)) \tag{1.1}$$

Los instrumentos de registro sísmico comúnmente usados son el sismógrafo y el acelerógrafo.

El *sismógrafo* se caracteriza por su alta sensibilidad, es decir, tiene la capacidad de ampliar decenas o cientos de miles de veces la velocidad con que se mueve el suelo, ya sea a causa de un sismo cercano muy pequeño o de uno grande pero lejano. A los registros obtenidos con este instrumento se les llama sismogramas.

Los *acelerógrafos* tienen la característica, a diferencia de los sismógrafos de registrar la aceleración del suelo, generalmente son capaces de registrar aceleraciones mayores que la gravedad terrestre, por lo que los registros obtenidos o acelerogramas nunca se encuentran saturados aun en sismos de gran escala.

1.3.1. Acelerógrafo. Roca[6] precisa que independientemente del tipo de registro que un acelerógrafo genere, analógico o digital, consta de tres etapas comunes:

Transductor de aceleración. Comúnmente denominado acelerómetro, es el detector de movimiento preferido para aplicaciones de supervisión de aceleración, puesto que su funcionamiento está directamente relacionado con esta variable física. Este instrumento es útil para mediciones tanto de muy baja como de muy alta frecuencia, pudiendo

[6]ROCA, Op. cit., p. 145.

encontrarse comercialmente una amplia variedad que cubre prácticamente cualquier tipo de aplicación, tanto para propósitos generales como para propósitos específicos.

Acondicionamiento de Señal. La información entregada por el acelerómetro, debe ser tratada dándole la forma apropiada para ser insertada dentro del registrador, por esta razón, el acondicionamiento de señales es un procesamiento importante que incluye funciones como: amplificación, filtrado y aislamiento eléctrico.

En la mayoría de las aplicaciones, según Pernia[7], esto significa cambiar la señal de salida del acelerómetro a un nivel de voltaje requerido, modificar el rango dinámico del sensor para maximizar la precisión del sistema de adquisición, eliminar las señales indeseables, limitar el espectro del sensor, acoplar la impedancia de salida del transductor con la impedancia de entrada del sistema de registro.

Adicionalmente, se puede realizar el procesamiento de las señales analógicas (tanto lineales como no lineales) para reducir la carga de proceso del sistema de adquisición de datos y el computador.

Etapa de Registro. Esta etapa es la encargada de almacenar los eventos que se producen. El registro como tal, puede ser de varios tipos:

- Registro gráfico directo: sobre película fotográfica.
- Registro en cinta magnética.
- Registro en memoria de estado sólido.

A partir de los registros obtenidos con el acelerógrafo, se trata de reproducir el movimiento del suelo o del punto de la estructura donde se halla el instrumento con la mayor precisión posible, los registros de aceleración permiten obtener la velocidad y el desplazamiento, así como realizar el análisis en frecuencia de las señales.

1.3.2. Acelerómetro. Los acelerómetros se definen como sensores que miden la aceleración lineal a lo largo de su eje sensible. Las técnicas convencionales para detectar y medir aceleración se basan en el principio descubierto por Isaac Newton, conocido

[7]PERNIA, Daniel. Introducción a la medición de vibración. Mérida, Venezuela: s.n. 2004, p. 2-6.

como la segunda ley de Newton, la cual de acuerdo a Thales[8], afirma que la fuerza neta aplicada sobre un cuerpo es proporcional a la aceleración que adquiere dicho cuerpo, su expresión matemática se muestra en la ecuación 1.2.

$$F = \frac{dp}{dt} \tag{1.2}$$

Donde p es la cantidad de movimiento, que se define como el producto de la masa de un cuerpo por su velocidad como en la ecuación 1.3.

$$p = m \cdot v \tag{1.3}$$

Entonces la segunda ley de Newton se expresa como en la ecuación 1.4.

$$\mathbf{F} = \frac{d(m \cdot \mathbf{v})}{dt} = m \cdot \frac{d\mathbf{v}}{dt} + \mathbf{v} \cdot \frac{dm}{dt} \tag{1.4}$$

Puesto que la masa sísmica es constante, dm/dt es igual a cero y la ecuación 1.4 se reduce a la ecuación 1.5, donde la fuerza es proporcional a la aceleración que sufre la masa m.

$$\mathbf{F} = m \cdot \mathbf{a} \tag{1.5}$$

Los acelerómetros convierten la aceleración de gravedad o de movimiento, en una señal eléctrica analógica proporcional a la fuerza aplicada al sistema, o mecanismo sometido a vibración o aceleración. Esta señal analógica indica, en tiempo real, la aceleración instantánea del objeto sobre el cual el acelerómetro está montado.

Los acelerómetros son direccionales, esto quiere decir que sólo miden aceleración en un eje. Para monitorear aceleración en tres dimensiones, se emplea acelerómetros multi-ejes (ejes *norte-sur (ns)*, *este-oeste (ew)*, *vertical (z)*), los cuales son ortogonales.

Los acelerómetros pueden ser pasivos o activos: los pasivos envían la carga generada por el elemento sensor (puede ser un material piezoeléctrico) y debido a que esta señal es muy pequeña, estos acelerómetros requieren de un amplificador para incrementar la señal.

Los acelerómetros activos incluyen circuitos internos para convertir la carga del acelerómetro a una señal de voltaje, pero requieren de una fuente constante de corriente para alimentar el circuito.

[8]THALES: SOCIEDAD ANDALUZA DE EDUCACIÓN MATEMÁTICA. Leyes de Newton [en línea]. Disponible en Internet: `http://thales.cica.es/rd/Recursos/rd98/Fisica/02/leyes.html`

Las opciones de salida eléctrica dependen del sistema utilizado con los acelerómetros, las opciones analógicas comunes son: voltaje, corriente, y frecuencia, las digitales son las señales paralelas y seriales. Otra opción es usar acelerómetros con una salida de cambio de estado de interruptores o alarmas.

Sensibilidad. La sensibilidad en un acelerómetro se define como la salida de voltaje o carga por unidad de entrada de aceleración. Esta salida se especifica por unidad de aceleración, usualmente en milivoltios ($1mV = 10^{-3}$ Voltios) si es en voltaje, o en picocoulombios ($1pC = 10^{-12}$ Coulombs) si es en carga. La unidad de aceleración comúnmente empleada es el múltiplo del valor de la aceleración de la gravedad de la Tierra, indicada por *"g"*, cuyo valor según el personal del laboratorio de física de la Universidad de Nariño, se ha estandarizado en $978cm/s^2$ para la ciudad de San Juan de Pasto.

Aunque un instrumento puede tener una sensibilidad "típica", ésta suele diferir. Las especificaciones de funcionamiento incluyen una tolerancia que se expresa como un porcentaje de variación, por ejemplo $100mV/g \pm 5\,\%$. Este ejemplo significa que la sensibilidad real está dentro del intervalo de 95 a $105mV/g$, lo cual se verifica cuando se encuentra en condiciones estables. Con el propósito de realizar mediciones confiables, se recomienda calibrar y verificar periódicamente el funcionamiento y la sensibilidad de los instrumentos.

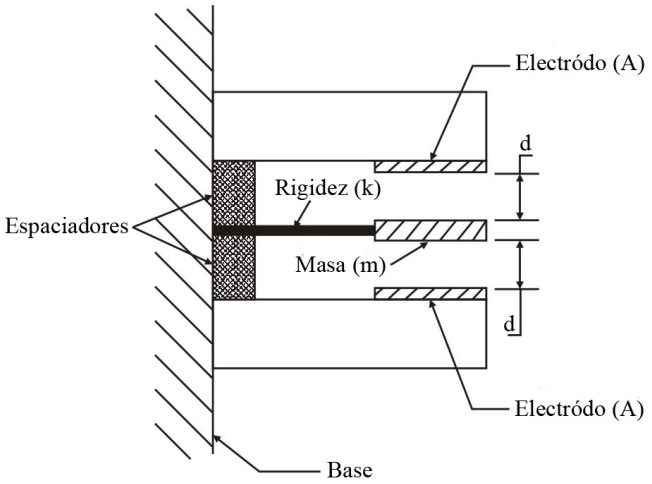
Tipos de Acelerómetros. Según Fernandez[9], existen diferentes tipos de acelerómetros basados en diversas tecnologías, algunos de ellos se describen a continuación.

Acelerómetros Capacitivos. Los acelerómetros capacitivos operan con una técnica donde la capacitancia del elemento sensor interno, cambia en proporción a la aceleración aplicada.

En la figura 1.3 se representa el elemento sensor, que consiste en dos placas conductoras paralelas tipo electródo con área de exposición A y una masa m suspendida por medio de un elemento con rigidez k. Entre la masa y los electródos existe una distancia base d simétrica, que se controla con precisión, por lo que el aire que existe en el hueco entre cada electródo y la masa sísmica forma un "capacitor mecánicamente variable".

[9]FERNÁNDEZ OLTRA, Rubén. Sistema de Adquisición de Posicionamiento Geográfico [en línea]. Disponible en Internet: `http://upcommons.upc.edu/pfc/bitstream/2099.1/4453/1/Sistema%20de%20Adquisici%C3%B3n%20de%20Posicionamiento%20Geogr%C3%A1fico.pdf` p. 18-26.

Figura 1.3. Estructura general de un acelerómetro capacitivo



Fuente: MAGGIOLO, Gustavo. Nota de Aplicación: detector sísmico con acelerómetro XYZ MMA7260Q. UTN - Facultad Regional Paraná (Argentina). Rev.N, 006/2007. Disponible en internet: Disponible en internet: `http://www.electrocomponentes.com.ar/educacion/pdf/programa/2007/resumenes/UTN_PARANA_NA_006_DETECTOR_SISMICO.pdf`. p. 3.

Cuando el elemento es acelerado, de acuerdo con la segunda ley de Newton, se presenta una fuerza inercial (F) proporcional a la aceleración que sufre la masa (m) como en la ecuación 1.5.

Acelerómetros Piezoeléctricos. Este tipo de acelerómetros aprovechan los fenómenos piezoeléctricos en algunos materiales, para generar una señal eléctrica proporcional a la aceleración de la vibración a la que son sometidos.

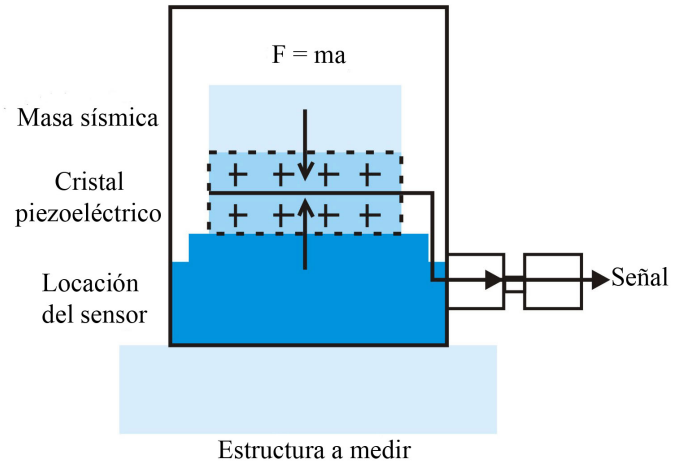
El elemento activo del acelerómetro es un cristal piezoeléctrico pegado a una masa conocida. Un lado del cristal está conectado a un poste rígido en la base del sensor; en el otro lado se encuentra adjunto un material llamado masa sísmica, como se observa en la figura 1.4.

Cuando el acelerómetro se encuentra sometido a vibración se genera una fuerza que actúa sobre el elemento piezoeléctrico y es igual al producto de la aceleración por la masa sísmica. Debido al efecto piezoeléctrico se genera una salida de carga proporcional a la fuerza aplicada.

Al medir la corriente de salida se calcula la aceleración: directamente si se trata de un acelerómetro de salida de corriente (coulombio/g) o convirtiéndola a un voltaje de baja

impedancia si se trata de un acelerómetro de salida de voltaje.

Figura 1.4. Diagrama de un acelerómetro Piezoeléctrico



Fuente: NATIONAL INSTRUMENTS. Tutorial: Acondicionamiento de Señales [en línea]. Disponible en Internet: `http://digital.ni.com/worldwide/latam.nsf/87e62f4c89ea9df9862564250075e6e4/a6c5283ceb7366cc86256e5900705e37/\protect\T1\textdollarFILE/Acondicionamiento%20de%20Se%C3%B1ales.pdf`. p. 4.

Algunas desventajas de este tipo de dispositivos es que no permiten para una entrada común mantener una señal de salida constante y, además la frecuencia de trabajo de los mismos no puede ser demasiado elevada.

Acelerómetros desarrollados con Tecnología Micromecánica (MEMS). Se denomina MEMS (Sistemas Micro Electro-Mecánicos) a una tecnología de base que se utiliza para crear dispositivos diminutos. Este tamaño puede oscilar entre pocas micras pudiendo llegar hasta un milímetro de diámetro.

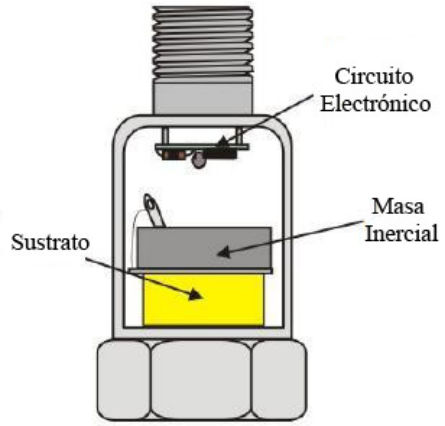
Estos dispositivos pueden fabricarse como cualquier tipo de semiconductor a partir de láminas de silicio o de vidrio, aunque existen métodos más avanzados de fabricación.

Acelerómetros Piezo-resistivos. Estos dispositivos basan su funcionamiento en la propiedad que tienen las resistencias eléctricas de cambiar su valor cuando el material se deforma mecánicamente, ese cambio depende del tipo de material y de como fue dopado.

A diferencia del piezo-eléctrico utiliza un sustrato en lugar de un cristal piezo-eléctrico, como se ejemplifica en la figura 1.5. En esta tecnología, la fuerza que ejerce la masa sobre el sustrato varía su resistencia, la cual forma parte de un circuito que mide la

intensidad de la corriente. La ventaja de esta tecnología respecto a la piezo-eléctrica es que puede medir aceleraciones hasta cero Hz de frecuencia.

Figura 1.5. Diagrama de un acelerómetro piezo-resistivo



Fuente: CARACENA, Teófilo. Instrumentación Industrial: Acelerómetros [en línea]. Disponible en Internet: http://74.125.47.132/search?q=cache:B9OrwglNRPQJ:www.gii.upv.es/personal/gbenet/IIN/treballs%25200607%C3+Industrial:+Acelerometros.&cd=1&hl=es&ct=clnk&gl=co.p.5.

Acelerómetros Mecánicos. Estos acelerómetros utilizan bobinas, imanes, resortes y una masa para medir aceleración. Son usados en sistemas que originan movimientos oscilatorios cuando se encuentran sometidos a aceleración (servoacelerómetros).

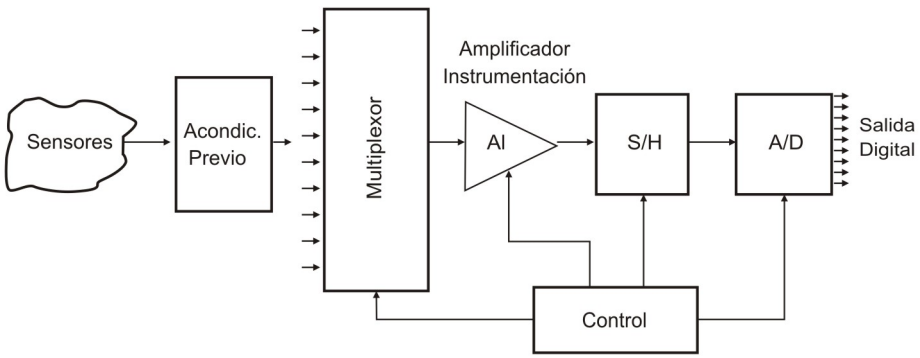
2 SISTEMA DE ADQUISICIÓN DE DATOS (SAD)

Para llevar a cabo un análisis eficaz sobre un determinado proceso, se hace necesario captar una serie de datos para posteriormente analizarlos, tratarlos y almacenarlos.

Los datos o variables que se captan tienen un carácter analógico, mientras que su tratamiento, almacenamiento y análisis son mucho más eficaces cuando se hace digitalmente; por esta razón se debe llevar a cabo una transformación de los datos desde el campo analógico al campo digital, lo cual implica la implementación de una serie de módulos electrónicos.

Al conjunto de los diferentes módulos electrónicos que permiten llevar a cabo la transformación anterior se le denomina Sistema de Adquisición de Datos (SAD), siendo su estructura general la mostrada en la figura 2.1.

Figura 2.1. Diagrama de bloques de un SAD genérico



Fuente: Tema 3. Adquisición de datos [en línea]. Disponible en Internet: `http://www.depeca.uah.es/wwwnueva/docencia/IT-INF/ctr-eco/Tema3.pdf.p.1.`

2.1. BLOQUES PRINCIPALES

Según Rubia[1], los bloques principales de un SAD son:

Sensores o transductores. Son los encargados de convertir la variable física a medir en señal eléctrica. Esta señal eléctrica suele ser de muy bajo nivel, por lo que generalmente se requiere un acondicionamiento previo, consiguiendo así niveles de tensión/corriente adecuados para el resto de los módulos del SAD.

Acondicionamiento previo. Es la etapa encargada de filtrar y adaptar la señal proveniente del transductor a la entrada del conversor analógico/digital que transforma la señal analógica en digital. Este proceso se encarga de:

- Adaptar el rango de salida del transductor al rango de entrada del conversor (normalmente en tensión).
- Acoplar la impedancia de salida del transductor con la impedancia de entrada del conversor.

La adaptación tiene como objetivo: aprovechar el margen dinámico del conversor, de modo que la máxima señal de entrada debe coincidir con el nivel máximo que soporta el convertidor.

Por otro lado, el acoplamiento de impedancias es imprescindible ya que los transductores presentan una salida de alta impedancia, que normalmente no puede excitar la entrada de un conversor, cuya impedancia típica suele estar entre 1 y 10 k$Omega$.

Multiplexor. Este circuito se encarga de seleccionar la señal de entrada que va a ser tratada en cada momento. En el caso de que solamente deseáramos tratar con una única señal, este circuito no es necesario.

Amplificador de instrumentación. La función de este bloque es amplificar la señal de entrada del SAD para que su margen dinámico se aproxime lo máximo posible al margen dinámico del conversor A/D (ADC) consiguiéndose de esta forma máxima resolución. En un SAD con varios canales de entrada, cada canal tendrá un rango de entrada distinto, por lo que es necesario que este amplificador sea de ganancia programable.

[1] RUBIA, Juan. Nociones básicas sobre adquisición de señales [en línea]. Versión 1, diciembre de 2000. Disponible en Internet: `http://www.redeya.com/electronica/tutoriales/adatos/adquisicion.html`

S & H (*Sample & Hold*, Muestreo y Retención). Este circuito es el encargado de tomar la muestra del canal seleccionado (*sample*) y mantenerla (*hold*) durante el tiempo que dura la conversión. Este circuito será necesario siempre que la señal de entrada sufra variaciones apreciables durante el tiempo que dura la conversión. Si el ADC posee su propio circuito S & H, no será necesario añadirlo a su entrada.

Conversor Análogo - Digital (Conversor A/D). Se encarga de realizar la conversión analógico/digital, proporcionando un código digital de salida que representa el valor de la muestra adquirida en cada momento. Es uno de los módulos fundamentales en cualquier SAD y sus características pueden condicionar al resto de los módulos/circuitos del sistema.

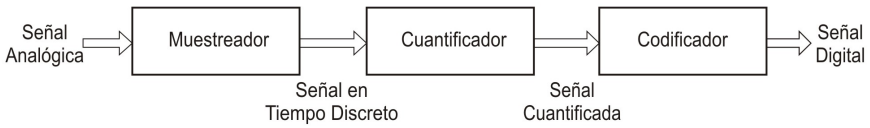
Las características básicas que se tienen en cuenta en un conversor A/D son:

- Rango de entrada: rango de voltajes que soporta el dispositivo en la entrada.
- Impedancia de entrada: impedancia medida del conversor en sus terminales de entrada.
- Número de bits: número de bits que tiene la palabra de salida del conversor, y por tanto, es el número de pasos que éste admite. Así, un conversor de 8 bits sólo podrá dar a la salida $2^8 = 256$ valores posibles.
- Resolución: mínimo valor que puede distinguir el conversor en su entrada analógica, o dicho de otro modo, la mínima variación, V_i, en el voltaje de entrada que se necesita para cambiar en un bit la salida digital.
- Tiempo de conversión: tiempo que tarda el conversor en realizar una medida y que depende de la tecnología empleada; proporciona la máxima frecuencia de la señal a medir. Este tiempo se mide como el transcurrido desde que el conversor recibe una señal de inicio de conversión hasta que en la salida aparece un dato válido.

Según Pallás[2], en la conversión analógica/digital intervienen tres procesos: muestreo, cuantificación y codificación, como se observa en la figura 2.2.

[2]PALLÁS, Ramón. Adquisición y distribución de señales. Barcelona : Marcombo, 1993. p. 255-277.

Figura 2.2. Procesos de la conversión A/D



Fuente: MORALES, Johnny. Introducción al Procesamiento de Señales y al Análisis de Sistemas de Variable Discreta [en línea]. Disponible en Internet: `http://electronica.udea.edu.co/cursos/circuitos3/20071/Pres1.pdf`.

Muestreo. Consiste en tomar muestras periódicas de la amplitud de onda. La velocidad con que se toma esta muestra, es decir, el número de muestras por segundo, es lo que se conoce como *frecuencia de muestreo.*

El muestreo de la señal implica pérdida de información respecto a la señal de entrada, ya que de un número infinito de valores posibles para la entrada, sólo se tiene un valor finito de valores posibles para la salida; debido a esto, es fundamental saber cuántas muestras se toman, lo cual depende del error medio admisible, el método de reconstrucción de la señal y el uso final de los datos de la conversión.

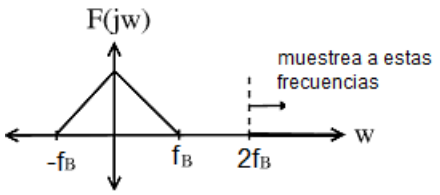
La precisión de los datos muestreados puede mejorarse: aumentando el número de muestras por ciclo, filtrando antes del multiplexado o filtrando la salida del conversor digital/analógico.

El objetivo fundamental de la adquisición es poder reconstruir la señal muestreada de una manera fiel, por lo tanto, se tiene en consideración *el teorema de Nyquist* o *teorema de muestreo*, el cual plantea que la frecuencia mínima de muestreo para poder reconstruir la señal debe ser el doble de la frecuencia de la señal a medir.

Si la señal está limitada en banda en $[-f_B, f_B]$, se puede reconstruir perfectamente de sus muestras $x_s[n] = x(nT)$, para una frecuencia de muestreo $f_S = \frac{2\pi}{T} > 2f_B$, donde f_B es la frecuencia más alta en la señal. En la figura 2.3 se ilustra la frecuencia de Nyquist $f_N = 2f_B$.

Las muestras tomadas son retenidas por un circuito el tiempo suficiente para permitir evaluar su nivel (cuantificación). Desde el punto de vista matemático este proceso no se contempla, ya que se trata de un recurso técnico debido a limitaciones prácticas, y carece de modelo matemático.

Figura 2.3. Ilustración de la frecuencia de Nyquist



Fuente: ROMBERG, Justin. Nyquist Theorem [en línea]. Disponible en internet: `http://cnx.org/content/m10791/2.4`

Cuantificación. Durante este proceso se mide el nivel de tensión de cada una de las muestras obtenidas en el proceso de muestreo y se les atribuye un valor finito (discreto) de amplitud, seleccionado por aproximación dentro de un margen de niveles previamente fijado.

Los valores preestablecidos para ajustar la cuantificación se eligen en función de la propia resolución que utilice el código empleado durante la codificación. Si el nivel obtenido no coincide exactamente con ninguno, se toma como valor el inferior más próximo.

Así pues, la señal digital que resulta tras la cuantificación es sensiblemente diferente a la señal eléctrica analógica que la originó, por lo que siempre va a existir una cierta diferencia entre ambas, que es lo que se conoce como error de cuantificación, el cual se produce cuando el valor real de la muestra no equivale a ninguno de los escalones disponibles para su aproximación y la distancia entre el valor real y el que se toma como aproximación es muy grande.

Codificación. La codificación es el paso en el que se asigna un código binario a la señal resultado de la cuantificación, de modo que las etapas posteriores al conversor puedan leer estos datos adecuadamente.

Etapa de Salida. Es el conjunto de elementos que permiten conectar el sistema de adquisición de datos con el resto del equipo y puede ser desde una serie de buffers digitales incluidos en el circuito conversor, hasta una interfaz RS232, RS485 o Ethernet, entre otros, para conectar a un computador o estación de trabajo, en el caso de sistemas de adquisición de datos comerciales.

2.2. PARÁMETROS CARACTERÍSTICOS DE UN SAD

Los parámetros que caracterizan a un SAD son básicamente tres:

- Número de canales: depende del número de señales a adquirir.
- Exactitud de la conversión: viene impuesta por los circuitos utilizados, es decir, multiplexores, amplificadores, *S&H* y *ADC*, esencialmente.
- Velocidad de muestreo: este parámetro especifica la velocidad a la que el SAD puede adquirir y almacenar muestras de las entradas. Las muestras pertenecerán a un único canal o a varios, según la configuración.

En general, la velocidad se debe identificar con el número de muestras por unidad de tiempo que pueden obtenerse de un canal.

2.3. IMPLEMENTACIÓN DEL SISTEMA DE ADQUISICIÓN DE MICROSISMOS

El sistema de adquisición de microsismos se compone de diversos dispositivos que permiten la medición de vibraciones de baja frecuencia del suelo.

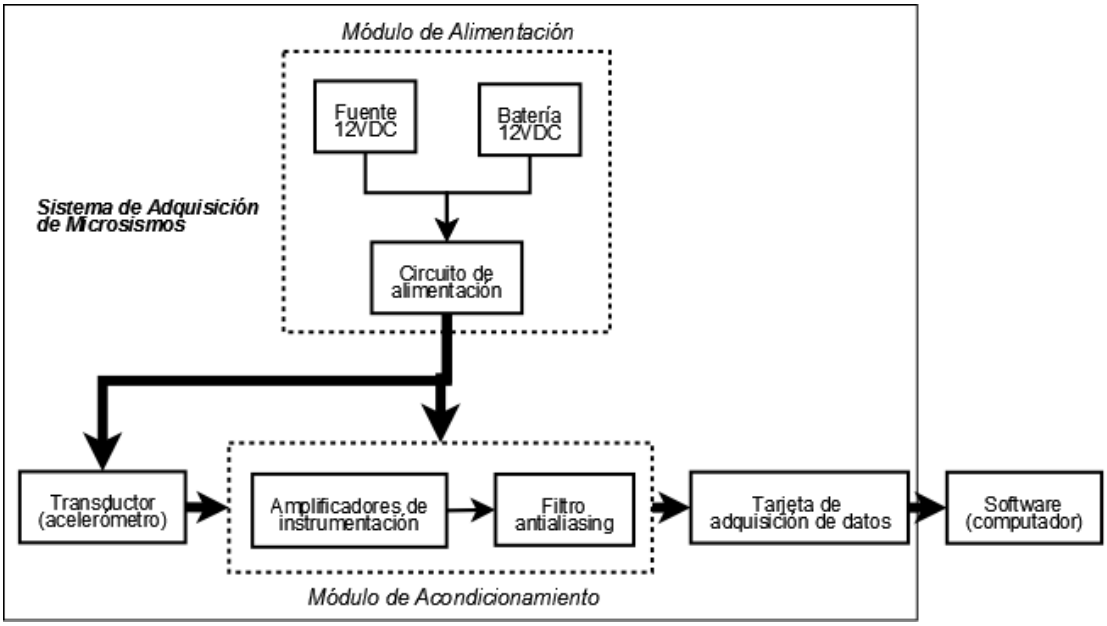
En el diagrama de bloques de la figura 2.4 se muestra cada uno de los módulos que forman parte del sistema desarrollado.

2.3.1. Módulo de alimentación. Se utiliza para alimentar el transductor de aceleración y el módulo de acondicionamiento de señales; está compuesto por:

- Fuente dual de corriente directa con salida de ±12 voltios.
- Batería recargable de 12 Voltios.
- Tarjeta o circuito impreso de alimentación.

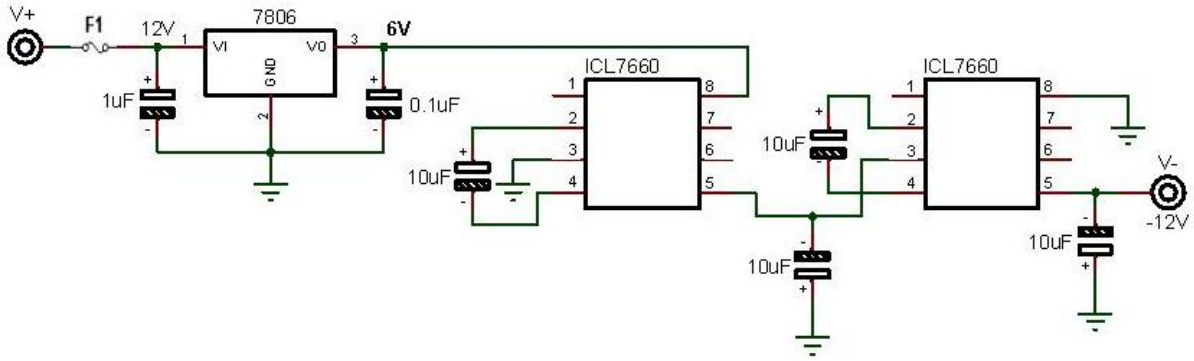
El equipo se enciende por medio de un interruptor *On/Off* y se utiliza un relé de estado sólido que permite la alimentación del sistema mediante su conexión a la red eléctrica por medio de la fuente dual o utilizando la batería recargable.

Figura 2.4. Diagrama de bloques del sistema de adquisición de microsismos



La etapa de acondicionamiento se alimenta con ±12 voltios. Para obtener un voltaje negativo a partir de la batería se implementa el circuito conversor de la figura 2.5.

Figura 2.5. Circuito conversor de voltaje a −12 voltios



Fuente: INTERSIL, ICL7660 Data Sheet [en línea]. Disponible en internet: `www.intersil.com/data/fn/fn3072.pdf`

Los 12 voltios de entrada se convierten a un voltaje negativo usando el circuito integrado conversor de voltaje ICL7660 en configuración cascada, que permite un voltaje de salida

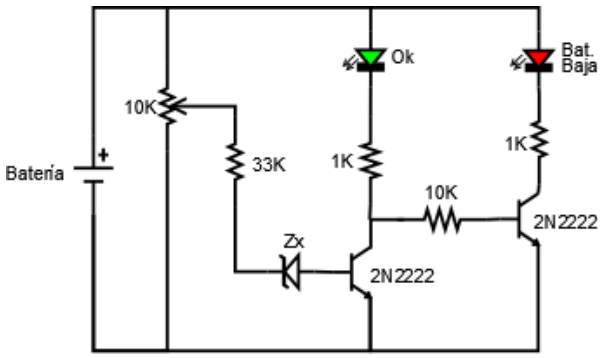
como se indica en la ecuación 2.1, en donde n hace referencia al número de integrados utilizados.

$$V_{salida} = -n \cdot V_{entrada} \tag{2.1}$$

Mediante el regulador de voltaje positivo 7806, se obtienen 6 voltios a la entrada del primer ICL7660, para conseguir -12 voltios a la salida del segundo ICL7660 utilizado.

Se dispone de un fusible (F1) de 500 miliamperios, entre la batería y el circuito como protección contra sobre corrientes y corto circuitos, esto no es necesario para la fuente dual puesto que tiene una protección integrada. Para conocer el estado de carga de la batería, se utiliza el circuito indicador que se observa en la figura 2.6.

Figura 2.6. Circuito indicador de estado de carga para la batería

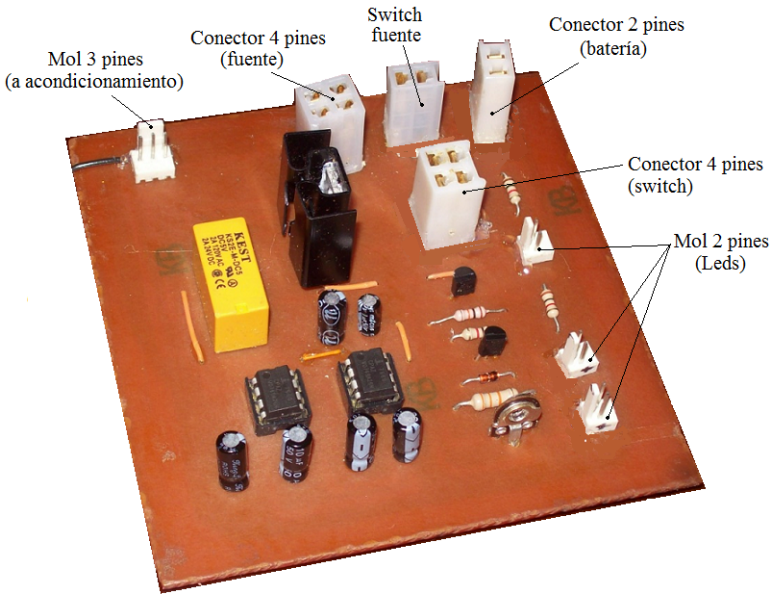


Se diseña una tarjeta que integra los circuitos electrónicos correspondientes a este módulo como se muestra en la figura 2.7.

2.3.2. Transductor de aceleración. El transductor que se utiliza es el acelerómetro triaxial marca Crossbow modelo CXL04GP3-R, diseñado para proporcionar valores de aceleración en los ejes X, Y y Z, por medio de un nivel de tensión proporcional a cada una de ellos.

Es un acelerómetro capacitivo micromaquinado de silicio, con recubrimiento de nylon, posee un cable de 2 metros de longitud altamente flexible con un conector hembra de 5 pines, cuya funcionalidad se describe en el cuadro 2.1. En la figura 2.8 se observa su aspecto externo.

Figura 2.7. Tarjeta de alimentación



Cuadro 2.1. Diagrama de pines del acelerómetro CXL04GP3-R

Pin	Color	Función
1	Rojo	Alimentación
2	Negro	Tierra
3	Blanco	Salida Eje-X
4	Amarillo	Salida Eje-Y
5	Verde	Salida Eje-Z

CROSSBOW TECHNOLOGY, CXL-GP Series Data Sheet [en línea]. Disponible en Internet: `http://www.hoskin.qc.ca/uploadpdf/Instrumentation/CrossBow/hoskin_GP%20Series_47e00ece030de.pdf` p. 2.

Este sensor juega un papel primordial en el sistema dadas sus características particulares que le dan una gran autonomía, incluso para trabajar con un consumo mínimo. Ofrece un amplio rango dinámico, tiene una excelente respuesta en frecuencia e incluye internamente un regulador de voltaje que le permite ser alimentado con una fuente no regulada entre 5.5 y 36 voltios.

La señal de salida es acondicionada internamente mediante microcircuitos electrónicos que convierten el cambio de capacitancia interna en una señal de voltaje, de acuerdo a la aceleración aplicada.

Algunos de sus valores característicos se indican en el cuadro 2.2.

Figura 2.8. Acelerómetro triaxial Crossbow CXL04GP3-R

Cuadro 2.2. Características del acelerómetro CXL04GP3-R.

Especificaciones	**Valor**
Rango de entrada (g)	±4
Sensibilidad (V/g)	
Eje X	0.498
Eje Y	0.499
Eje Z	0.500
Voltage Zero-G	(V)
Eje X	2.382
Eje Y	2.368
Eje Z	2.374
Ruido (mg rms)	10
Ancho de Banda (Hz)	DC - 100
Rango de temp. de operación (°C)	-40 a +85
Tamaño (cm)	1.98 x 4.45 x 2.72
Peso (gm)	46

CROSSBOW TECHNOLOGY, CXL-GP Series Data Sheet [en línea]. Disponible en Internet: `http://www.hoskin.qc.ca/uploadpdf/Instrumentation/CrossBow/hoskin_GP%20Series_47e00ece030de.pdf` p. 2.

La ecuación 2.2 permite encontrar el voltaje de salida V_s para cada uno de los ejes del acelerómetro, donde G_v es la ganancia introducida por el amplificador de instrumentación que se explica en la etapa de acondicionamiento previo; el factor de escala F_s es la sensibilidad en $voltios/g$; a es el valor de la aceleración en g; y el voltaje offset V_{off} es el voltaje de salida en zero-G, es decir, cuando el sensor está en reposo. Estos valores son proporcionados en la hoja de calibración del sensor y se indican en el cuadro 2.2.

$$V_s = G_v \cdot ((F_s \cdot a) + V_{off}) \tag{2.2}$$

En la selección de este acelerómetro se tienen en cuenta los requisitos mínimos para la instalación de acelerógrafos de la norma NRS-98, Título A, Cápítulo A-11, en donde se

establece que el sensor debe ser triaxial (dos componentes horizontales y una vertical), con una escala total de al menos el 100 % de la aceleración de la gravedad (1G), la frecuencia natural de aproximadamente 50Hz, un amortiguamiento crítico del 70 % y rango dinámico de 135dB en su respuesta plana, la cual está al menos en la banda de 0.1 a 20Hz.

2.3.3. Módulo de acondicionamiento. Las señales provenientes de los tres ejes del acelerómetro son captadas por el módulo de acondicionamiento, en el cual se realiza una amplificación y un filtrado previos a la etapa de conversión análogica-digital.

Amplificadores de instrumentación. Debido a que existe la necesidad de medir señales muy pequeñas del orden de pocos milivoltios en presencia de perturbaciones, se utilizan amplificadores de instrumentación para el acondicionamiento de las señales de salida del acelerómetro.

Un amplificador de instrumentación es un dispositivo creado a partir de amplificadores operacionales, diseñado para tener una alta impedancia de entrada que evita atenuaciones o distorsiones en la señal y un alto rechazo de modo común que ayuda a un bajo nivel de ruido; presenta una ganancia finita, precisa y estable.

Debido a que el sensor es triaxial, se utilizan tres amplificadores de instrumentación *INA128*, para cada uno de las ejes de salida. Estos se configuran para obtener una ganancia de $3V/V$, la cual se fija conectando una resistencia externa R_G igual a $25k\Omega$, entre los pines 1 y 8 del circuito integrado, valor calculado con la ecuación 2.3.

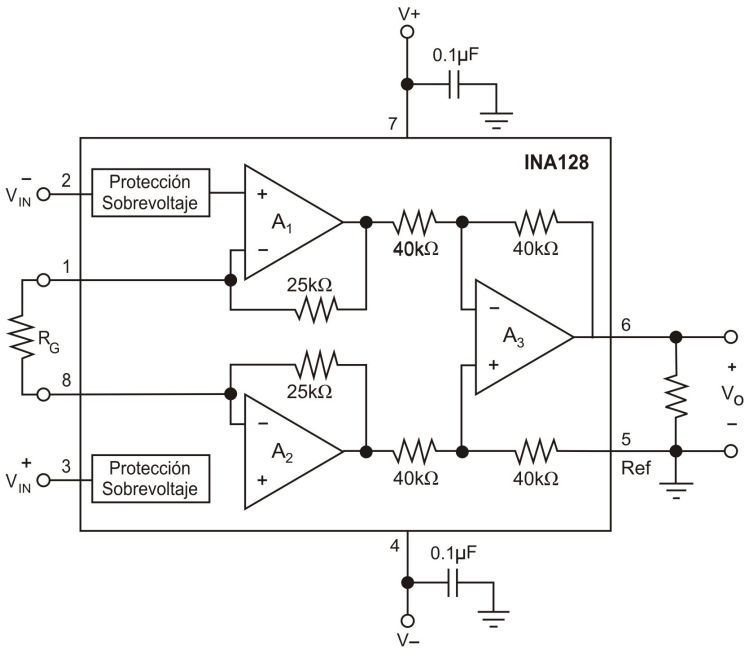
$$R_G = \frac{50k\Omega}{G-1} \tag{2.3}$$

La configuración utilizada para realizar las conexiones básicas del INA128 se observan en la figura 2.9.

A la entrada diferencial positiva (pin 3) llega la señal de salida del acelerómetro, la entrada diferencial negativa (pin 2) se conecta a un capacitor y a una resistencia en paralelo conectados a tierra, para evitar voltajes en modo común que puedan generar ruido adicional.

Filtro anti-aliasing. En el proceso de conversión analógico digital se toman muestras de la señal de entrada. Si la señal contiene información en frecuencias superiores a la

Figura 2.9. Conexiones básicas del amplificador de instrumentación INA128



Fuente: BURR-BROWN, INA128 precision, low power instrumentation amplifiers [en línea]. U.S.A., 1996. Disponible en internet: `http://www.datasheetcatalog.org/datasheet/texasinstruments/ina128.pdf.p.8.`

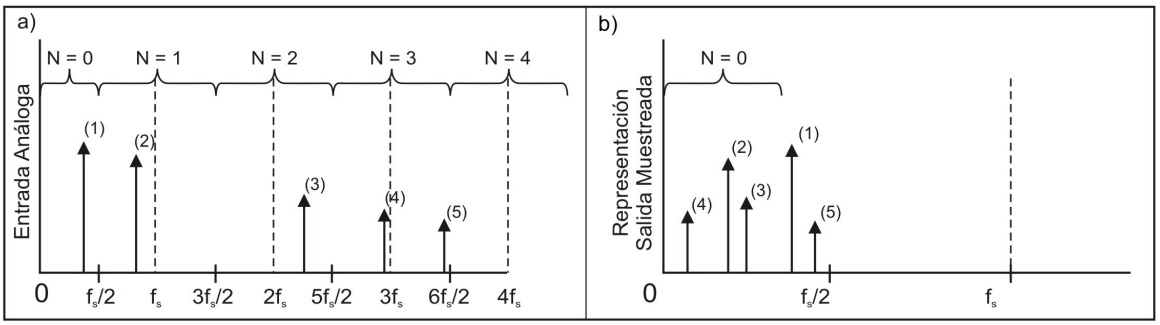
mitad de la frecuencia de muestreo, la toma de muestras no se hará de manera correcta, lo cual conlleva a una interpretación inexacta de la señal, apareciendo una distorsión o un "alias", que genera desfases o desplazamientos temporales de la señal.

La frecuencia máxima de la que se puede tomar muestras es la frecuencia de Nyquist y es igual a la mitad de la tasa de muestreo.

En todos los sistemas de procesamiento de señales digitales, la tasa de muestreo debe ser suficientemente superior a la frecuencia de Nyquist, para evitar el efecto aliasing. Este fenómeno en el dominio de la frecuencia se ejemplifica en la figura 2.10.

En las dos partes de la figura 2.10, el *eje x* identifica la frecuencia de muestreo f_s, en la parte izquierda se observan cinco segmentos en la banda de frecuencia, el segmento $N = 0$ tiene un intervalo desde DC a la mitad de la tasa de muestreo, en este ancho de banda, el sistema registra confiablemente el contenido de frecuencia de la señal análoga de entrada.

Figura 2.10. Efecto Aliasing



BAKER, Bonnie. Anti-aliasing, analog filters for data acquisition systems. Disponible en internet: `http://ww1.microchip.com/downloads/en/AppNotes/00699b.pdf`. p. 4.

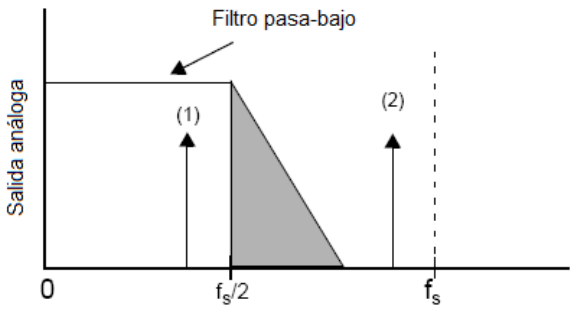
En los segmentos donde $N > 0$, se puede llegar a creer que la señal tiene una frecuencia mucho menor de la que realmente tiene, de modo que pueden aparecer señales de alta frecuencia superpuestas, como ruido y otras senoidales que aparentemente no son ruido, pero que también afectan a la señal bajo medida en el ancho de banda del segmento $N = 0$, como se observa en la gráfica b de la figura 2.10. Matemáticamente las frecuencias altas se superponen con la ecuación 2.4.

$$f_{alias} = |f_{in} - Nf_s| \tag{2.4}$$

Para eliminar o reducir significativamente el aliasing, el sistema de adquisición incluye un filtro pasa-bajas antes de la conversión análogo/digital, este concepto se ilustra en la figura 2.11.

En este diagrama, el filtro pasa-bajas atenúa la segunda porción de la señal de entrada a la frecuencia 2 de la gráfica. Se observan dos regiones, la región a la izquierda de la figura está dentro del ancho de banda de DC a $f_s/2$, la región sombreada es la banda de transición del filtro. Debido a que esta región es mayor que $f_s/2$, las señales dentro de esta banda de frecuencia producirán distorsiones en la salida del sistema de muestreo. Los efectos de este error pueden ser minimizados moviendo la frecuencia de corte del filtro a un valor menor a $f_s/2$ o incrementando el orden del filtro, en ambos casos, la mínima ganancia del filtro a $f_s/2$ debe ser menor que la relación señal a ruído (SNR) del sistema de muestreo.

Figura 2.11. Filtro pasa-bajo para eliminar efecto aliasing



Fuente: BAKER, Bonnie. Anti-aliasing, analog filters for data acquisition systems. Disponible en internet: `http://ww1.microchip.com/downloads/en/AppNotes/00699b.pdf`. p. 5.

Para realizar el diseño del filtro antialias se tienen en cuenta los siguientes parámetros:

- El ancho de banda de la señal de entrada tiene un rango de DC a $50Hz$.
- La frecuencia de muestreo a utilizar es de $200Hz$.
- El conversor A/D de la tarjeta de adquisición de datos a utilizar es de 16 bits tipo SAR, que tiene una SNR ideal igual a $98dB$, por lo tanto, el filtro debe atenuar la señal a aproximadamente $-98dB$ en $100Hz$.
- La señal análoga será únicamente filtrada sin invertirla ni introducir ganancia.

Se utiliza el programa FilterLab Versión 2.0 de Microchip Technology, útil para el diseño y optimización de filtros teniendo en cuenta las anteriores consideraciones. El diseño del filtro anti-aliasing resultante es un filtro Butterworth de 8° orden, en configuración Sallen - Key, en el cual se utilizan cuatro amplificadores operacionales y ocho capacitores que denotan el orden del filtro y que se ubican en la entrada y en la rama de realimentación del amplificador, como se observa en la figura 2.12. Este filtro activo ofrece la ventaja de proveer aislamiento entre las etapas, tomando ventaja de la alta impedancia de entrada y la baja impedancia de salida de los amplificadores operacionales.

La respuesta en frecuencia obtenida en el diseño del programa se muestra en la figura 2.13.

Figura 2.12. Diseño del circuito del filtro anti-aliasing Butterworth de 8° orden, con frecuencia de corte $F_C = 50Hz$, configuración Sallen - Key.

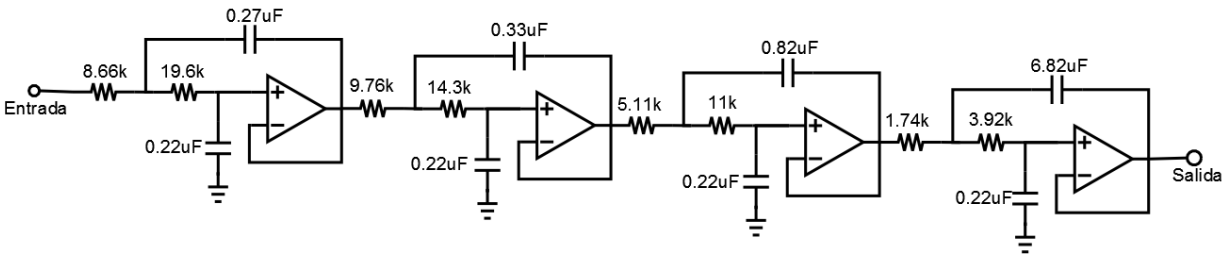
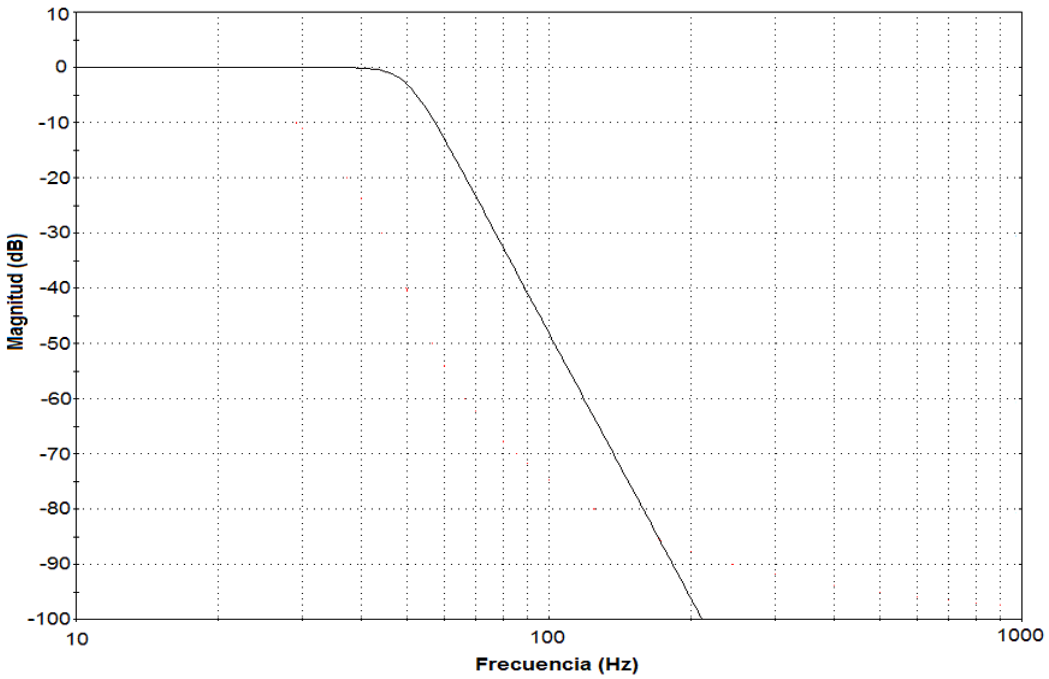


Figura 2.13. Respuesta en frecuencia del diseño del filtro Butterworth anti - aliasing de 8° orden y 50 Hz de frecuencia de corte.



Al ser un filtro Butterworth, se caracteriza por tener una respuesta plana con las mínimas oscilaciones hasta la frecuencia de corte, lo cual implica que la salida se mantiene constante en la banda de paso.

Este filtro se implementa para cada salida del acelerómetro, utilizando el circuito inte-

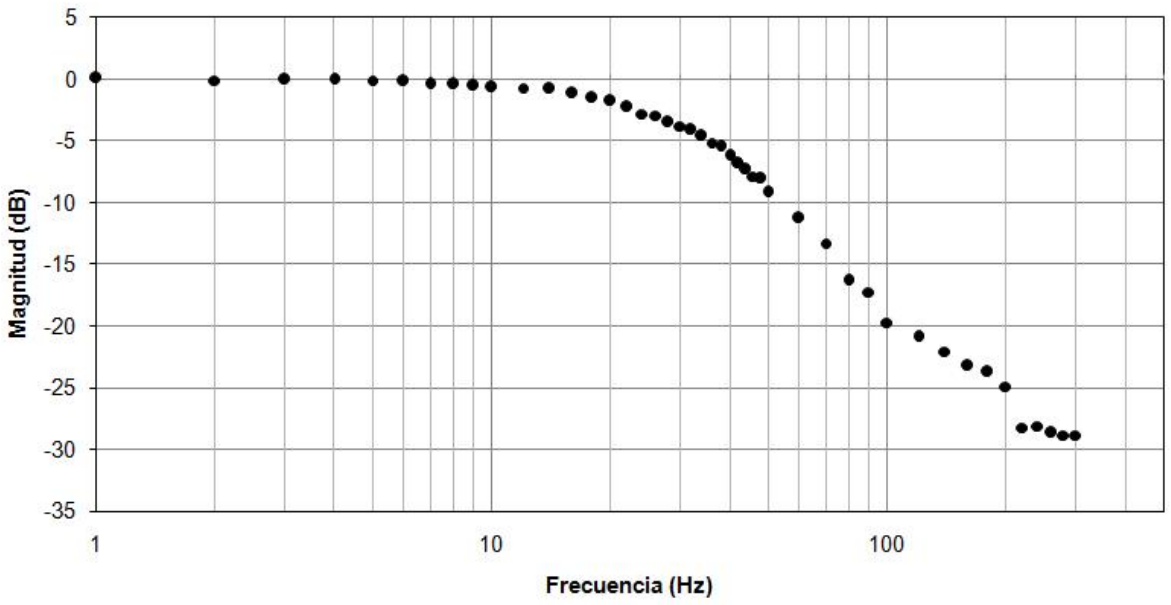
grado TL084 compuesto por 4 amplificadores operacionales y valores de resistencias y capacitores cercanos a los del circuito de diseño.

Para realizar la prueba de este filtro se utiliza una onda senoidal con amplitud de 2Vpp a la entrada, proporcionada por un generador de señales marca Protek 9205C. Se hace un barrido de frecuencias desde $1Hz$ hasta $300Hz$, registrando los valores de la señal de entrada y de la señal de salida obtenida para cada frecuencia, los datos obtenidos se muestran en el cuadro 2.3, con estos valores es posible calcular la atenuación de la señal para cada frecuencia con la ecuación 2.5.

$$A = 20 \times log(\frac{Vs}{Ve}) \tag{2.5}$$

Con los datos del cuadro 2.3 se grafica frecuencia en Hertz versus la atenuación en dB para obtener la respuesta en magnitud del filtro anti-alias, como se indica en la figura 2.14.

Figura 2.14. Respuesta en frecuencia del filtro Butterworth anti - aliasing de octavo orden y 50 Hz de frecuencia de corte.



La etapa de amplificación y la etapa de filtrado se implementan en una tarjeta como se

Cuadro 2.3. Datos de prueba obtenidos a partir del filtro antialias implementado

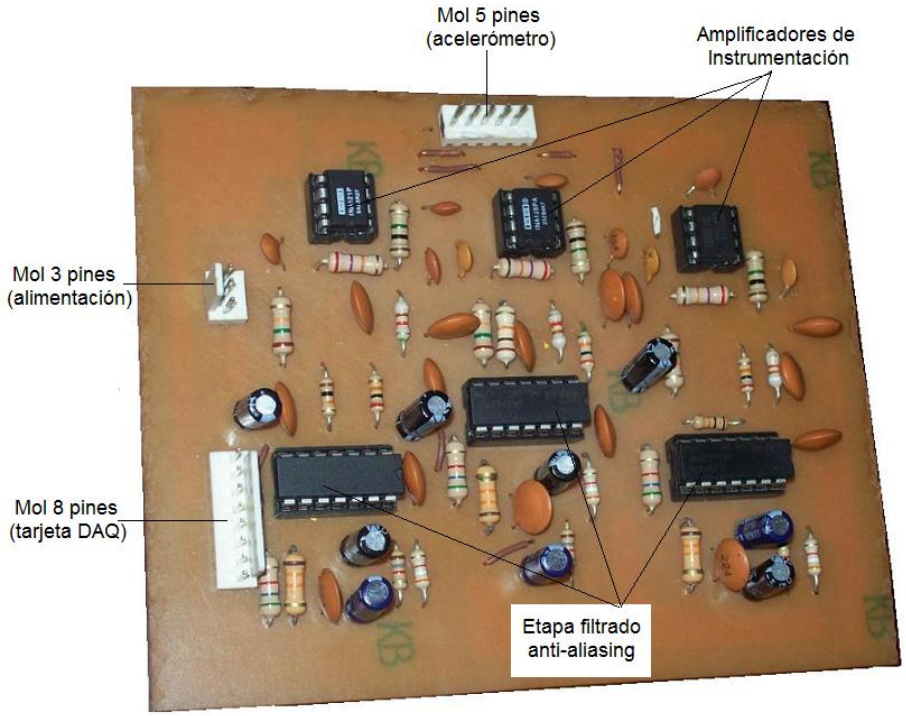
Voltaje entrada(V)	**Voltaje salida(V)**	**Frecuencia(Hz)**	**Atenuación(dB)**
1.070	1.087	1	0.1418
1.071	1.060	2	-0.0897
1.119	1.120	3	0.0023
1.094	1.095	4	0.0032
1.077	1.068	5	-0.0761
1.046	1.031	6	-0.1263
1.067	1.019	7	-0.3997
1.021	0.974	8	-0.4024
1.045	0.986	9	-0.4996
1.050	0.973	10	-0.6573
1.045	0.964	12	-0.6948
1.027	0.937	14	-0.7979
1.029	0.907	16	-1.0958
1.025	0.863	18	-1.4947
1.030	0.839	20	-1.7821
1.032	0.794	22	-2.2777
1.033	0.747	24	-2.8118
1.022	0.732	26	-2.9025
1.023	0.686	28	-3.4753
1.022	0.662	30	-3.7705
1.024	0.637	32	-4.1278
1.024	0.603	34	-5.1482
1.026	0.550	38	-5.4140
1.025	0.502	40	-6.1962
1.022	0.466	42	-6.8276
1.027	0.444	44	-7.2790
1.022	0.414	46	-7.8482
1.025	0.405	48	-8.0684
1.024	0.357	50	-9.1501
1.024	0.280	60	-11.2792
1.020	0.221	70	-13.2719
1.021	0.157	80	-16.2706
1.019	0.140	90	-17.2311
1.017	0.105	100	-19.7053
1.019	0.092	120	-20.8929
1.017	0.080	140	-22.0504
1.018	0.071	160	-23.1886
1.016	0.066	180	-23.7373
1.016	0.058	200	-24.8869
1.021	0.040	220	-28.2056
1.016	0.040	240	-28.0716
1.015	0.038	260	-28.6273
1.014	0.037	280	-28.8064
1.015	0.037	300	-28.8149

indica en la figura 2.15.

Después de desarrollar la etapa de acondicionamiento previo, se procede a adquirir la señal mediante la tarjeta de adquisición de datos.

2.3.4. Tarjeta de adquisición de datos. Se utiliza la tarjeta de adquisición de datos NI USB-9215 de la empresa National Instruments.

Figura 2.15. Tarjeta módulo de acondicionamiento



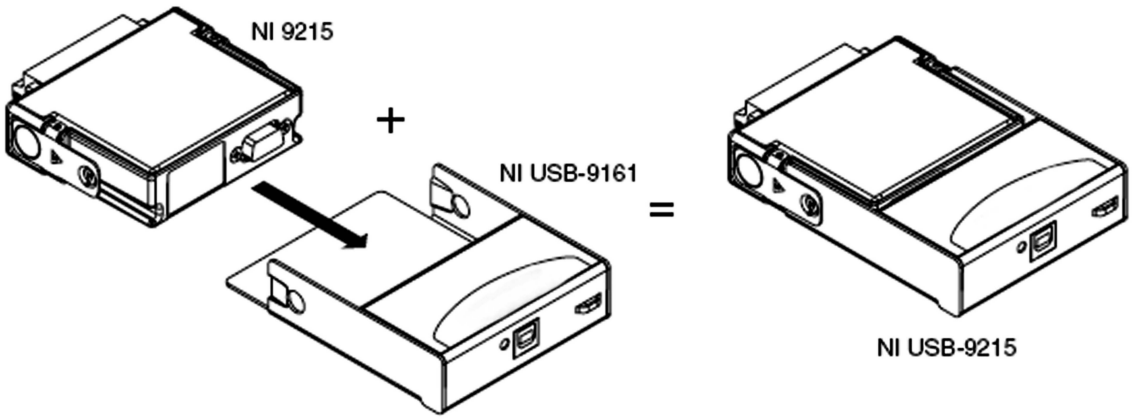
Provee una interfaz USB para cuatro canales de entrada análoga, muestreo simultáneo de 16 bits y acondicionamiento de señal integrado.

Consiste en dos componentes: el módulo de entrada analógica compactRIO 9215 y un módulo adaptador o portador USB 9161. El circuito del módulo es tal que no permite compatibilidad con otros portadores y está diseñado para un controlador específico: NIDAQmxBase para la tarjeta USB 9215. El módulo de entrada se inserta en el portador como se indica en la figura 2.16.

Conexiones. Cada canal tiene dos terminales de tornillo o pin central para la entrada positiva y negativa de una señal diferencial. En este caso como la señal proveniente del acondicionamiento es unipolar, se conecta al terminal positivo del canal y el terminal negativo se conecta a tierra, como en la figura 2.17.

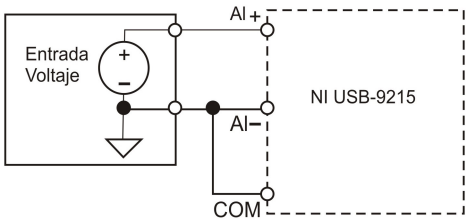
En el cuadro 2.4 se observa el diagrama del módulo de entrada de la tarjeta NI USB-9215 y la asignación de sus terminales.

Figura 2.16. Componentes tarjeta NI USB-9215: Portador NI USB9161 y módulo Compact-RIO NI9215.



Fuente: NATIONAL INSTRUMENTS CORPORATION, Operating Instructions USB-9215 [en línea], 2004. Disponible en internet: `ftp://ftp.ni.com/support/manuals/371261a.pdf`. p. 2.

Figura 2.17. Conexión de señales de voltaje no diferenciales a la entrada de la tarjeta



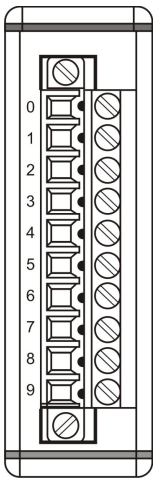
Fuente: Ibid., p. 6.

Teniendo en cuenta lo anterior, la señal del *eje X* se conecta al terminal 0, la del eje *Y* al terminal 2 y la del *eje Z* al terminal 4, los terminales 1, 3 y 5 se conectan a tierra, al igual que el terminal 9 o común (*COM*).

Circuito interno. La tarjeta tiene internamente protección sobre voltaje, un amplificador de instrumentación para cada uno de los canales y un conversor análogo digital (ADC) de 16 bits. En la figura 2.18 se observa el diagrama de los bloques principales del circuito interno que hace parte de cada canal de la tarjeta NI USB-9215.

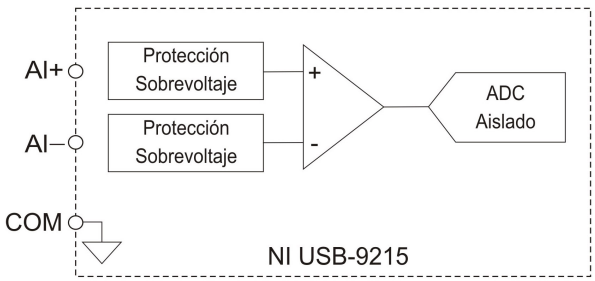
Los canales tienen amplificadores *track & hold* independientes, estos retienen la señal al mismo tiempo permitiendo el muestreo de los cuatro canales simultáneamente, lo cual es muy útil para conservar las relaciones de fase entre canales.

Cuadro 2.4. Terminales del módulo de entrada de la tarjeta de adquisición de datos NI USB-9215

MÓDULO	TERMINAL	SEÑAL
	0	AI0+
	1	AI0–
	2	AI1+
	3	AI1–
	4	AI2+
	5	AI2–
	6	AI3+
	7	AI3–
	8	No conectar
	9	COM

Fuente: NATIONAL INSTRUMENTS CORPORATION, Op. cit., p. 5.

Figura 2.18. Circuito interno de un canal en la tarjeta NI USB-9215



Fuente: Ibid., p. 7.

Especificaciones. En el cuadro 2.5 se describen las especificaciones típicas para un rango de temperatura de 0 a 60°C.

El conversor ADC tiene la capacidad de convertir una muestra analógica entre ±10 voltios y tiene un número máximo de 16 bits de salida, lo cual permite determinar el número máximo de combinaciones de la salida digital, dado por: 2^n donde n es el número de bits.

Cuadro 2.5. Especificaciones de la tarjeta de adquisición de datos NI USB-9215

Característica	**Valor**
Canales de entrada	4
Resolución del ADC	16 bits
Tipo de ADC	SAR
Voltaje de entrada	±10V
Protección sobre voltaje	±30V
Máxima tasa de muestreo	
Canal 0	20kS/seg
Canal 1	10kS/seg
Canal 2	6.67kS/seg
Canal 3	5kS/seg
Tiempo de conversión	
Canal 0	$4\mu s$
Canal 1	$6\mu s$
Canal 2	$8\mu s$
Canal 3	$10\mu s$

Fuente: NATIONAL INSTRUMENTS CORPORATION, Op. cit., p.5.

La resolución entendida como el voltaje necesario (señal analógica) para lograr que en la salida (señal digital) haya un cambio del bit menos significativo (LSB) es igual a:

$$R = \frac{ViFS}{2^n - 1} \tag{2.6}$$

Donde $ViFS$ es el voltaje de entrada al convertidor para obtener una conversión máxima (todas las salidas son "1").

$$R = \frac{20V}{2^{16}} = 3{,}052 \times 10^{-4} \tag{2.7}$$

Lo cual quiere decir que por cada $0{,}3052mV$ que aumente el nivel de tensión entre las entradas del conversor, éste aumentará en una unidad su salida (siempre sumando en forma binaria bit a bit).

Es importante destacar que la interfaz USB brinda al sistema de adquisición de datos, alta velocidad con una tasa de transferencia de hasta 480 Mbps (60 MB/s), excelente confiabilidad y flexibilidad para ser usado tanto en computadores de escritorio, en computadores portátiles o en computadores industriales.

2.4. DESARROLLO DEL PROGRAMA DE ADQUISICIÓN Y REGISTRO

La tarjeta de adquisición de datos NI USB-9215 está diseñada para trabajar en los sistema operativos Linux y MacOs. Para utilizarse en Windows, es necesario instalar el controlador NI-DAQmx Base 2.2 versión descargable de la página web de la empresa National Instruments.

En cuanto se instala el controlador, el computador reconoce los dos dispositivos conectados, tanto el módulo adaptador USB, como el módulo de entrada analógica y los habilita para funcionar en Labview y para ser manipulados a través de la aplicación DataLogger de National Instruments. Además ofrece compatibilidad con el lenguaje de programación C incluyendo bibliotecas que declaran y definen las diferentes funciones que permiten realizar la adquisición.

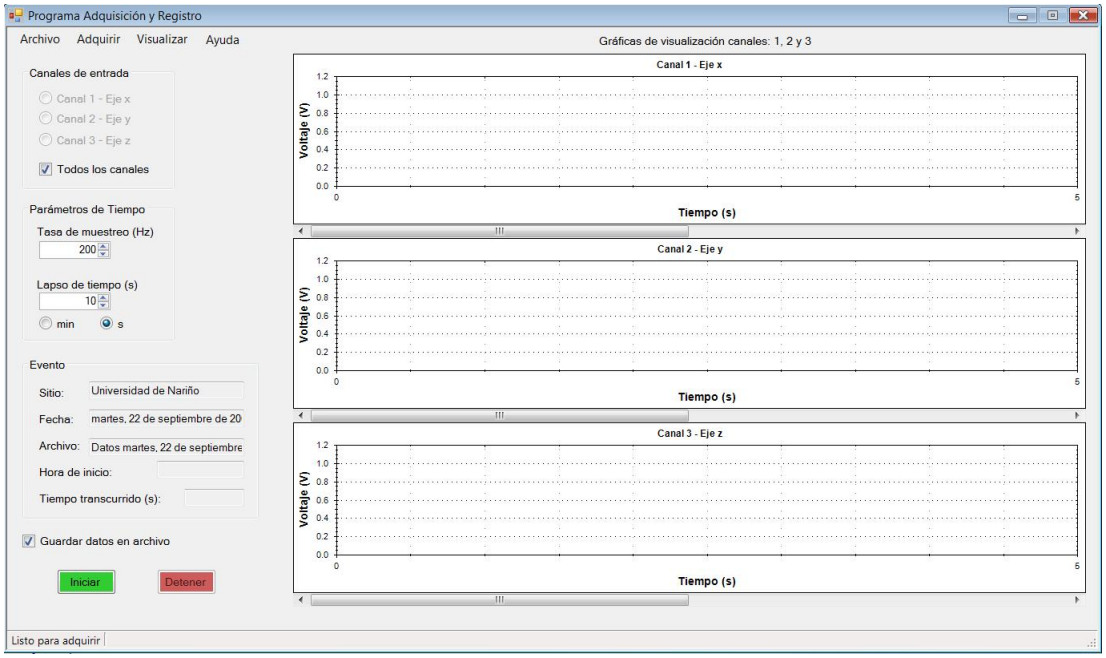
La programación de la interfaz de usuario para la adquisición, se desarrolla mediante el programa Visual C++ 2008 Express Edition, un entorno de desarrollo integrado (IDE) especialmente diseñado para la elaboración y depuración de código escrito en la interfaz de programación de aplicaciones de Microsoft Windows. Esta versión es gratuita y se puede descargar desde la página web de Microsoft.

Visual C++ 2008 Express proporciona numerosas características de nivel profesional y permite crear aplicaciones y componentes para Windows extremadamente eficaces. Ofrece características que ayudan a optimizar el proceso de programación de software con C++, un lenguaje orientado a objetos de propósito general derivado del C, que se adapta a múltiples situaciones. Incorpora las extensiones administradas para C++ aprovechando las ventajas de Microsoft .NET, como la administración de recursos, tipos unificados y componentes en modo remoto. Con .NET se logra una mejora en la gestión de memoria para la recolección sin problemas de elementos no utilizados y una menor complejidad de los programas.

También proporciona los nuevos formularios Windows Forms que son compatibles con cualquier lenguaje basado en .NET, permitiendo realizar gran parte de las tareas mediante asistentes y simplificando la creación de aplicaciones basadas en Windows, centralizando en formularios primarios la lógica común y la interfaz de usuario para toda la solución.

2.4.1. Diseño de la interfaz gráfica. El programa de adquisición y registro de datos se desarrolla usando la plantilla de los formularios Windows Forms con sus herramientas gráficas como botones, cuadros de texto, menús, cuadros de diálogo, entre otros controles. Se diseña la ventana principal que se observa en la figura 2.19.

Figura 2.19. Ventana principal del programa de adquisición y registro



Elementos de la ventana principal. La ventana principal de la interfaz de usuario tiene las siguientes partes:

Barra de título. Indica el título del programa. Tiene la función de controlar la ventana permitiendo minimizar o cerrar la aplicación.

Barra de menús. Permite el acceso a los siguientes menús:

- Menú Archivo. Contiene la opción de salir del programa.
- Menú Adquirir. Permite iniciar la adquisición y cambiar la información general del evento a registrar mediante un cuadro de diálogo.
- Menú Visualizar. Contiene submenús para seleccionar los canales a visualizar y las variables de los ejes; el eje de las abscisas puede ser tiempo o número de muestras y el eje de las ordenadas, aceleración o voltaje.

- Menú Ayuda. Despliega el archivo de ayuda en el cual se describen las opciones del programa y el modo de uso.

Control de canales de entrada. Permite escoger el canal mediante el cual se desea realizar la lectura o adquisición de datos, estos pueden ser: canal 1 donde se conecta el eje x del sensor, canal 2 donde ingresa el eje y del sensor, el canal 3 correspondiente al eje z del sensor o todos los canales, para observar simultáneamente el registro de los tres ejes del acelerómetro.

Control de parámetros de tiempo. Posibilita seleccionar la frecuencia a la cual se muestrean las señales de entrada y fijar el lapso de tiempo que corresponde a la duración del registro de las señales en segundos o minutos.

Información del evento. Incluye el nombre del sitio o ubicación donde se realizan las mediciones, la fecha, el nombre del archivo donde se guardan los datos adquiridos, la hora de inicio y el tiempo trancurrido.

Control para guardar datos. Permite guardar un registro de los datos adquiridos.

Botones de control de la adquisición: Iniciar y Detener. Útiles para empezar la adquisición y para detenerla cuando sea necesario.

Barra de estado. Está situada en la parte inferior de la ventana, proporciona información adicional sobre el proceso de adquisición.

Gráficas. Permiten visualizar en pantalla las tres componentes de la señal microsísmica durante la adquisición. Se utiliza una biblioteca de clases de versión gratuita descargable desde internet, conocida como ZedGraph, que permite realizar diferentes tipos de gráficas y sirve para el propósito de visualizar la lectura de datos. Se accede a las funciones de esta biblioteca usando un control del cuadro de herramientas de Visual C++, el cual aparece al agregar el archivo ZedGraph.dll como parte de las referencias del proyecto.

2.4.2. Rutinas de la interfaz gráfica. Con el diseño de la interfaz de usuario, se procede a desarrollar el código correspondiente para realizar las funciones encargadas de cumplir el proceso de adquisición y registro de las señales microsísmicas. El programa está compuesto por los siguientes archivos que se observan dentro del explorador de soluciones en el entorno de desarrollo Visual C++:

- Archivos de código fuente.
- Archivos de encabezado.
- Archivos de recursos.

Dentro de los **archivos de código fuente** se encuentran:

- ClaseAdquisición.cpp: archivo de código fuente que contiene una clase denominada Adquisición, donde se definen las funciones necesarias para la lectura de datos mediante la tarjeta NI USB-9215.
- PanelInfoEvento.cpp: archivo de código fuente del cuadro de diálogo que contiene la información general de los eventos a registrar.
- AssemblyInfo.cpp: contiene la información general sobre el ensamblado, la cual se controla mediante un conjunto de atributos.
- stdafx.cpp: archivo de código fuente que contiene sólo las inclusiones estándar de bibliotecas.
- InterfazAdquisición.cpp: archivo de proyecto principal. Crea la ventana principal y la ejecuta.

Los **archivos de encabezado** son:

- ClaseAdquisición.h: archivo de encabezado de la clase Adquisición, donde se declaran las funciones para la adquisición de datos mediante la tarjeta de adquisición de datos NI USB-9215.
- Form1.h: incluye el código y el diseñador de la ventana principal de la interfaz de usuario.
- PanelInfoEvento.h: contiene el código y el diseñador del cuadro de diálogo de la información general.
- stdafx.h: archivo de inclusión de los archivos estándar o específicos de un proyecto, utilizados frecuentemente pero rara vez modificados. Aquí se mencionan los encabezados adicionales que el programa necesita.

- resource.h

Pasos para la adquisición continua de datos en múltiples canales mediante la tarjeta NI USB-9215. Con el driver instalado NI-DAQmx Base 2.2, se incluyen dos bibliotecas necesarias para programar las rutinas de adquisición de datos mediante la tarjeta NI USB-9215. Estos archivos son:

- El archivo de encabezado NIDAQmxBase.h: contiene las declaraciones de constantes, variables y funciones que permiten ejecutar rutinas para la adquisición de datos mediante la tarjeta en el lenguaje C++.
- El archivo nidaqmxbase.lib: en donde se definen las funciones o clases para ser ejecutadas. Cuando el programa intenta llamar o ejecutar lo que hay en el archivo de encabezado, debe conocer qué hay, dónde está, y cómo se llama. Este fichero .lib le dice al enlazador exactamente esa información, la cual debe introducir en el ejecutable, para que en el momento de llamar a una determinada función dentro de una biblioteca, éste sepa que hacer.

Para depurar y generar el código, Visual C++ necesita conocer la ruta de estos archivos que se encuentran en la carpeta del driver y especificar al vinculador la búsqueda de las llamadas a las funciones DAQmxBase que se hagan dentro de la programación en el archivo nidaqmxbase.lib, por lo tanto, se adiciona el archivo de encabezado en la página de propiedades de la ventana dentro del campo de directorios de inclusión adicionales y también la sentencia *#include "NIDAQmxBase.h"* en el código fuente. Igualmente, se incluye el archivo .lib en el campo de directorios de bibliotecas adicionales y la sentencia *#pragma comment(lib, "nidaqmxbase.lib")* en el código fuente.

Dentro de la ventana principal se realiza la lógica de programación para adquirir datos continuamente y de manera simultánea a través de tres de los canales de entrada análogos de la tarjeta; para esto, de acuerdo a la National Instruments[3], se tiene en cuenta los siguientes pasos:

1. **Crear la tarea o *task*.** Esto se hace mediante la función *DAQmxBaseCreateTask* con un parámetro de entrada *taskName*, el nombre asignado a la tarea y un parámetro

[3]NATIONAL INSTRUMENTS CORPORATION, NI-DAQmx Base 2.x C Function Reference Help [en línea]. 1°edición, ago. 2006. Disponible en internet: `http://student.agh.edu.pl/~koskak/ksp2/docsource/daqmxbasecfuncindex.htm`

de salida *taskHandle*, un puntero de referencia a la tarea creada por esta función.

Declaración de la función:
*int32 DAQmxBaseCreateTask (const char taskName[ ], TaskHandle *taskHandle);*

2. **Crear los canales de entrada virtuales.** Se crean los canales de entrada análoga para medidas de voltaje y se adicionan al *task* que se especifica en el paso anterior con el puntero de referencia *taskHandle*.

Declaración de la función:
int32 DAQmxBaseCreateAIVoltageChan (TaskHandle taskHandle, const char physicalChannel[ ], const char nameToAssignToChannel[ ], int32 terminalConfig, float64 minVal, float64 maxVal, int32 units, const char customScaleName[ ]);

Tiene los siguientes parámetros de entrada:

- *taskHandle*: la tarea a la cual se adicionan los canales que esta función crea.
- *physicalChannel*: los nombres de los canales usados para crear los canales virtuales. Se especifica una lista o rango de los tres canales de la siguiente manera: Dev1/ai0, Dev1/ai1 y Dev1/ai2. Dev1 es el nombre que se le asigna a la tarjeta cuando es reconocida en el puerto USB.
- *nameToAssignToChannel*: NIDAQmxBase comúnmente ignora este parámetro.
- *terminalConfig*: la configuración del terminal de entrada para los canales, que puede ser la configuración por defecto *DAQmx_ Val_ Cfg_ Default (-1)* entrada referenciada a tierra no diferencial que es igual a *DAQmx_ Val_ RSE*, *DAQmx_ Val_ NRSE* entrada no diferencial no referenciada a tierra y *DAQmx_ Val_ Diff* modo diferencial.
- *minVal*: valor mínimo en voltios a medir.
- *maxVal*: valor máximo en voltios a medir.
- *units*: unidades de medida, siempre en voltios.
- *customScaleName*: se pasa *NULL* para este parámetro.

3. **Fijar los parámetros de tiempo para el muestreo.** Se establece la fuente de reloj de muestreo o *sample clock*, la tasa del reloj y el número de muestras a adquirir o generar. Adicionalmente, se define el modo de muestreo como continuo.

Declaración de la función:
int32 DAQmxBaseCfgSampClkTiming (TaskHandle taskHandle, const char source[ ], float64 rate, int32 activeEdge, int32 sampleMode, uInt64 sampsPerChanToAcquire);

Los parámetros que necesita como entrada son:

- *taskHandle*: el *task* usado en esta función.
- *source*: el terminal fuente del *sample clock*, puede usarse una fuente externa.
- *rate*: la tasa de muestreo en muestras por segundo.
- *activeEdge*: especifica en que flanco del reloj se adquieren las muestras. Puede ser en el flanco ascendente *DAQmx_ Val_ Rising* o en el descendente *DAQmx_ Val_ Falling*.
- *sampleMode*: especifica si el *task* adquiere o genera muestras continuamente para lo cual se utiliza *DAQmx_ Val_ ContSamps* o un número finito de muestras *DAQmx _ Val_ FiniteSamps*.
- *sampsPerChanToAcquire*: el número de muestras a adquirir para cada canal en el *task*, esto es necesario si el modo de muestreo se define como *DAQmx_ Val_ FiniteSamps*

4. **Configurar el *buffer* interno.** Permite designar un espacio de memoria temporal en el que se almacenan los datos, para evitar que el programa se quede en algun momento sin datos.

Declaración de la función:
int32 DAQmxBaseCfgInputBuffer (TaskHandle taskHandle, uInt32 numSampsPerChan);

Usa como parámetros de entrada a *taskHandle*, el *task* creado y *numSampsPerChan*, el número de muestras que el buffer puede guardar para cada canal en el *task*.

5. **Empezar la adquisición.** Se llama a la función *start* que cambia el estado del *task* para empezar a medir. El uso de esta función se requiere para todas las aplicaciones

NI-DAQmx Base.

Declaración de la función:
int32 DAQmxBaseStartTask (TaskHandle taskHandle);

Requiere como parámetro de entrada *taskHandle*, el *task* a ejecutar.

6. **Lectura de datos.** Se leen los datos del *task* creado y configurado con las anteriores funciones para los tres canales.

Declaración de la función:
*int32 DAQmxBaseReadAnalogF64 (TaskHandle taskHandle, int32 numSampsPerChan, float64 timeout, bool32 fillMode, float64 readArray[ ], uInt32 arraySizeInSamps, int32 *sampsPerChanRead, bool32 *reserved);*

Parámetros de entrada:

- *taskHandle*: el *task* del cual se leen las muestras.
- *numSampsPerChan*: el número de muestras por canal a leer. Si se fija este parámetro a -1 *(DAQmx_ Val_ Auto)*, NI-DAQmx Base determina cuantas muestras leer basándose el modo de adquisición de muestras del *task*, continuo o finito.
- *timeout*: la cantidad de tiempo (segundos) que la función espera para leer las muestras.
- *fillMode*: especifica si las muestras se intercalan o no, es decir, si los datos se organizan de una manera no contigua, esto para aumentar el rendimiento. Si se utiliza *DAQmx_ Val_ GroupByChannel* los datos se agrupan por canal (no intercalados) y se agrupan de forma intercalada usando la sentencia *DAQmx_ Val_ GroupByScan Number*.
- *arraySizeInSamps*: el tamaño del *array* o arreglo en muestras, dentro del cual los datos son leídos.
- *reserved*: reservado para un uso futuro. Se pasa *NULL* a este parámetro.

Esta función genera como salida los siguientes parámetros:

- *readArray*: el array donde se leen los datos, organizado de acuerdo al parámetro *fillMode*.

- *sampsPerChanRead*: el número real de las muestras leídas por cada canal.

7. **Detener la adquisición.** Se para el *task*, el cual retorna al estado en el que se encontraba antes de llamar a la función *DAQmxBaseStartTask*. Se requiere usar esta función en todas las aplicaciones NI-DAQmx Base.

Declaración de la función:
int32 DAQmxBaseStopTask (TaskHandle taskHandle);

8. **Limpiar el *task*.** Para ejecutar esta función, se debe: asegurar la detención del *task* con anterioridad y que se han liberado los recursos reservados por éste.

Declaración de la función:
int32 DAQmxBaseClearTask (TaskHandle taskHandle);

Todas estas funciones tienen un valor de retorno llamado *status* de tipo *int32*, el cual indica si en algún momento se produce un error; un valor igual a 0 indica que no hay error, un valor positivo indica una advertencia y un valor negativo un error. Esto permite generar mensajes de error para que puedan ser corregidos por el usuario, mediante la función *DAQmxBaseGetExtendedErrorInfo*, cuya declaración es:

int32 DAQmxBaseGetExtendedErrorInfo (char errorString[ ], uInt32 bufferSize);

Esta función retorna información específica del error por medio del parámetro de salida *errorString*, una variable de tipo *char*. Se proporciona una variable de entrada *bufferSize*, que hace referencia al tamaño en bytes de *errorString*.

Rutina desarrollada para la adquisición. En la ventana principal de la interfaz de usuario se dispone de un botón Iniciar o también la opción Iniciar del menú Adquirir, mediante el cual se comienza la adquisición y se realiza la lectura y grabación de datos.

Al hacer click sobre él, cuando se ha seleccionado la opción "Guardar datos en archivo" que se encuentra en la parte inferior de la ventana principal, se abre un cuadro de diálogo *Guardar como*, en el cual el usuario puede introducir el nombre del archivo y la carpeta donde se guardarán los datos del evento a registrar. Por defecto, al iniciar el programa por primera vez, se crea una carpeta llamada *Adquisición y Registro* dentro de la carpeta personal Mis documentos en la unidad de disco C.

Luego se comienza a ejecutar la rutina de adquisición continua de datos siguiendo los pasos explicados en la sección anterior. Es importante tener en cuenta que las bibliotecas NIDAQmxBase.h y nidaqmxbase.lib están escritas en ANSI C, la primera estandarización del lenguaje C, pero que pueden compilarse con un compilador de C++.

El mismo programa, en un fichero con extensión *.c puede ser convertido en un programa en C++ cambiando la extensión a *.cpp. C++ permite muchas más posibilidades que C, debido a esto, se crea dentro del programa una clase llamada *ClaseAdquisición*, para hacer que las bibliotecas puedan usarse dentro de Visual C++ .NET, que utiliza C++ administrado y sean compatibles con el Common Language Runtime o CLR. [0].

La clase *Adquisición* tiene un archivo de encabezado donde se declara la clase con sus miembros, es decir la lista de funciones y variables y un archivo de código fuente, donde se definen las funciones y se inicializan las variables. El código de estos archivos se puede observar en el anexo A.

En la ventana principal, se crea un objeto o instancia de esta clase con las siguientes sentencias:

```
Adquisición * adq;
adq = new Adquisición();
```

Dentro del evento click del botón Iniciar se tienen los siguientes comandos para preparar la tarjeta e iniciar la adquisición:

```
//Crea el task
adq->CreateTask();
//Llama a la función ElegirCanal
ElegirCanal();
//Crea los canales virtuales de entrada de voltaje
adq->CreateAIVoltageChan();
//Llama a la función ConfigurarParamTiempos()
ConfigurarParamTiempos();
//Configura el reloj de la tarjeta
adq->CfgSampClkTiming();
//Configura el buffer.
adq->CfgInputBuffer()
//Cambia de estado al task para empezar a adquirir
adq->StartTask();
```

La función *ElegirCanal* permite conocer el canal o canales de entrada elegidos por el usuario. La función *ConfigurarParamTiempos* define la tasa de muestreo y el lapso

[0]*CLR es el fundamento de .NET Framework. Es responsable de administrar la ejecución del código en tiempo de ejecución y proporciona servicios básicos como: compilación, administración de memoria y de subprocesos, ejecución de código, cumplimiento de seguridad de tipos y comprobación de la seguridad de código.

de tiempo que el reloj de la tarjeta debe tener en cuenta para realizar la lectura de datos, además fija los valores de los ejes horizontales de las gráficas de visualización. Al finalizar cada paso, las funciones poseen una rutina que revisa si no ha sucedido ningún error, de lo contrario, aparece un mensaje de advertencia o error.

Después se realiza la lectura y grabación de datos dentro de un ciclo *while*, el cual se ejecuta mientras la condición que se le establezca permanezca como verdadera; en el momento en que la condición se convierte en falsa el ciclo termina.

La condición del ciclo es que el tiempo sea menor que la variable *stopTime* que hace referencia al valor del lapso de tiempo escogido por el usuario. Dentro del *while* se llama a la función de lectura, la cual lee una cierta cantidad de datos por segundo dependiendo de la frecuencia de muestreo, esto lo hace continuamente hasta que la condición del ciclo sea falsa, es decir, se haya cumplido el tiempo de muestreo. Los datos leídos se van almacenando en un array para cada uno de los canales. Luego se crea un archivo de texto (.txt) que contiene: un encabezado predeterminado con la información general del evento y los datos adquiridos en unidades de voltaje. Este proceso se resume en el diagrama de flujo de la figura 2.20 y el cuerpo de instrucciones es el siguiente:

```
while(time(NULL)< stopTime)
{
    //LLama a la función de lectura para comenzar a adquirir
    adq->ReadAnalog();
    for(int i = 0; i < adq->pointsRead; i++)
    {
        datosCanal1->SetValue(adq->data[i*3+0],i);
        datosCanal2->SetValue(adq->data[i*3+1],i);
        datosCanal3->SetValue(adq->data[i*3+2],i);
    }

    //Si el usuario escoje unidades de aceleración
    //se realiza la conversión de voltaje a aceleración (mg)
    if(bEjeyAceleración == true)
    {
        for(int k = 0; k< adq->pointsRead; k++)
        {
             datosCanal1[k] = ((datosCanal1[k] - 6.658) / (2.7953*0.498))*1000;
             datosCanal2[k] = ((datosCanal2[k] - 6.718) / (2.8371*0.499))*1000;
             datosCanal3[k] = ((datosCanal3[k] - 8.044) / (2.7992*0.5))*1000;
        }
    }

    //Se filtran los datos leídos para evitar el ruido de 60Hz
    FiltroNotch(datosCanal1);
    FiltroNotch2(datosCanal2);
    FiltroNotch3(datosCanal3);

    //Crea el archivo donde se guardan los datos
    try
    {
         fileDatos = gcnew FileInfo(nombreArchivo);
         //Si ya existe adiciona los nuevos datos al final del archivo
         if(fileDatos->Exists == true)
         strDatos = fileDatos->AppendText();
         //Si no existe crea un archivo de texto nuevo
         else
```

```
        {
           strDatos = fileDatos->CreateText();
           //Aquí se escribe el encabezado del archivo
        }
        //Escribe en el archivo los datos que se adquieren (en voltaje)
        for (int i = 0; i < adq->pointsRead; i++)
        {
           strDatos->WriteLine("{0:g} {1:g} {2:g}",adq->data[i*3+0],
                     adq->data[i*3+1],adq->data[i*3+2]);
        }
     }
     __finally
     {
          //Cierra el objeto StreamWriter para liberar recursos
          strDatos->Flush();
          strDatos->Close();
     }
}
//Detiene el task
adq->StopTask();
adq->ClearTask();
```

Filtro Notch. Dentro del ciclo *while*, se utiliza un filtro rechaza banda para evitar el ruido de 60Hz en la adquisición, se diseña en Matlab para una frecuencia de muestreo de 200Hz, con las siguientes instrucciones:

```
%Frecuencia de muestreo
fs = 200;
%Frecuencias pasabanda normalizadas
Wp = [(55*2/fs) (65*2/fs)];
%Frecuencias de la banda de rechazo normalizadas
Ws = [(58*2/fs) (62*2/fs)];
%Máxima atenuación en la banda de paso (dB)
Rp = 3;
%Mínima atenuación en la banda de rechazo(dB)
Rs = 40;
%Selecciona el orden del filtro Butterworth a usar y
%la frecuencia de corte, dependiendo de Wp y Ws
[N,Wn]=buttord(Wp, Ws, Rp, Rs);
%Diseña un filtro rechaza banda de orden N,
%retorna coeficientes en B(numerador) y A(denominador)
[B,A] = butter(N, Wn, 'stop');
%Herramienta de visualización del filtro
fvtool(B,A)
```

Se obtienen los coeficientes de un filtro Butterworth de orden 6, usando la herramienta fvtool para visualizar el comportamiento del filtro se observa la respuesta en frecuencia que se muestra en la figura 2.21, en donde se tiene una magnitud de -3dB para las frecuencias 55.64 y 64,22Hz y una magnitud igual a -40dB en las frecuencias 58 y 62 Hz, lo cual implica la eliminación de los 60Hz característicos de la red eléctrica.

Figura 2.20. Diagrama de flujo del ciclo *while* para lectura de datos.

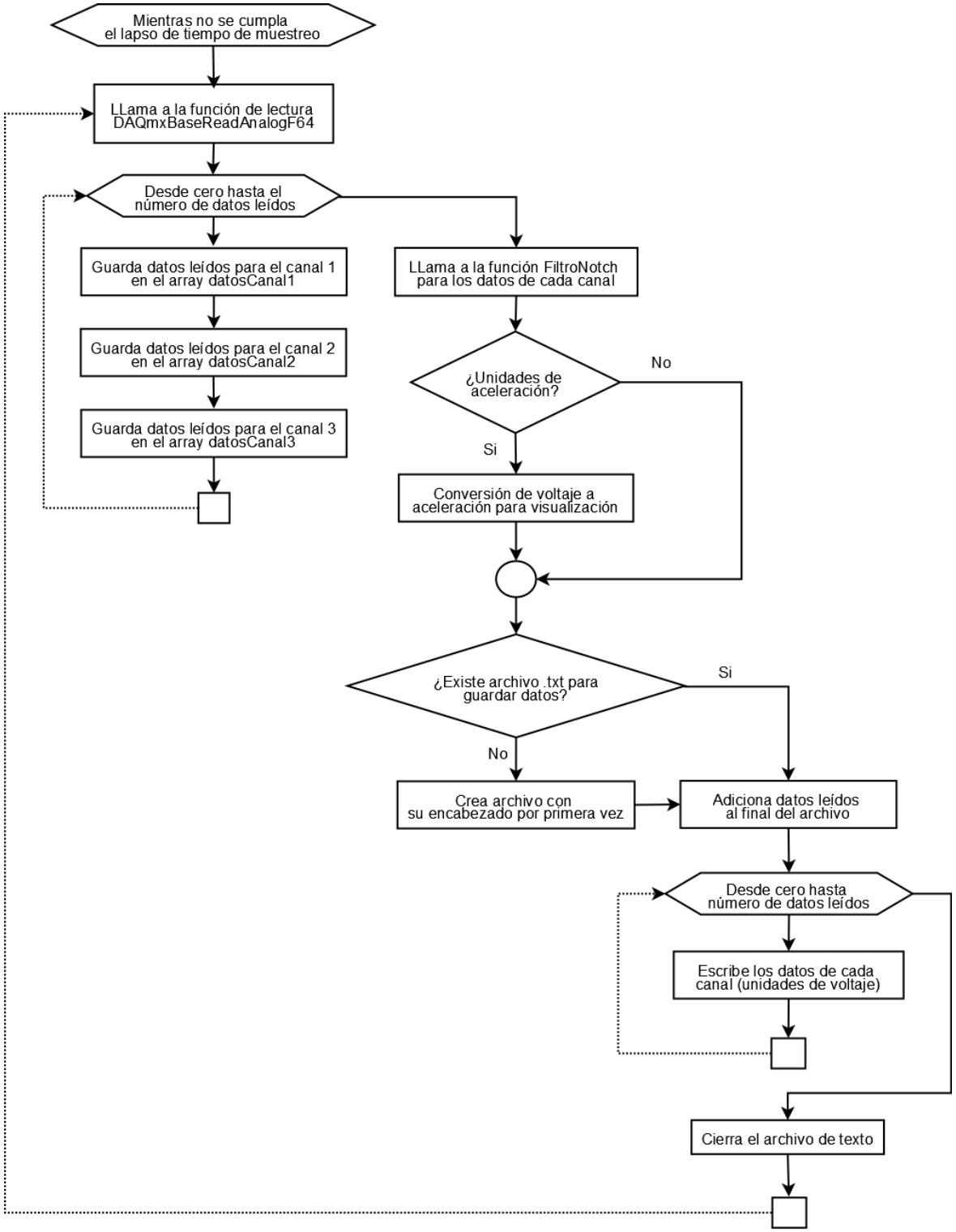
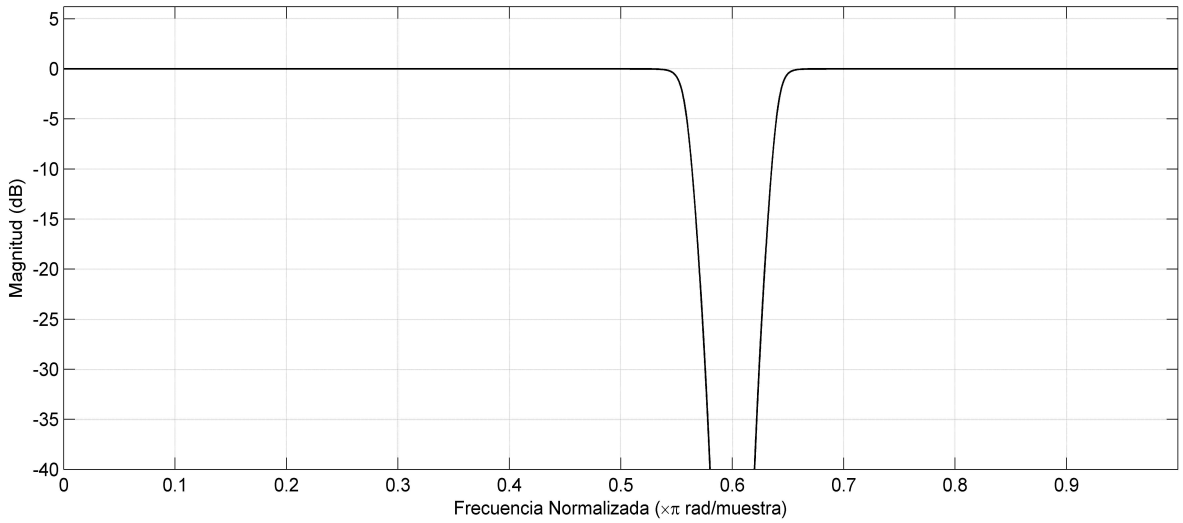


Figura 2.21. Respuesta en magnitud del filtro notch (dB) en diseño.



Para desarrollar la rutina de la función *FiltroNotch* en la ventana de adquisición, se utilizan los coeficientes obtenidos en matlab y el siguiente cuerpo de instrucciones:

```
private: Void FiltroNotch(array<double>^ df)
{
    #define NZEROS 12
    #define NPOLES 12
    #define GAIN   1.686639413e+00              //Ganancia para fs = 200Hz

    static double xv[NZEROS+1], yv[NPOLES+1];
    array<double>^ datosfiltro = gcnew array<double> (10000);

    for(int k = 0; k< adq->pointsRead; k++)
    {
         xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4]; xv[4] = xv[5];
         xv[5] = xv[6]; xv[6] = xv[7]; xv[7] = xv[8]; xv[8] = xv[9]; xv[9] = xv[10];
         xv[10] = xv[11]; xv[11] = xv[12];
         xv[12] = df[k] / GAIN;

         yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4]; yv[4] = yv[5];
         yv[5] = yv[6]; yv[6] = yv[7]; yv[7] = yv[8]; yv[8] = yv[9]; yv[9] = yv[10];
         yv[10] = yv[11]; yv[11] = yv[12];
         yv[12] = (xv[0] + xv[12]) + 3.7168023503 * (xv[1] + xv[11]) + 11.7560915460 * (xv[2] + xv[10])
                  +  23.3382905480 * (xv[3] + xv[9]) + 40.2332055110 * (xv[4] + xv[8])
                  +  51.9781811740 * (xv[5] + xv[7]) + 59.0107358460 * xv[6] + (-0.3515244284 * yv[0])
                  + (-1.4198733686 * yv[1]) + (-4.8681782816 * yv[2]) + (-10.5081362870 * yv[3])
                  + (-19.6863212650 * yv[4]) + (-27.7090035860 * yv[5]) + (-34.2724017790 * yv[6])
                  + (-32.9724037690 * yv[7]) + (-27.8755622870 * yv[8]) + (-17.7130256690 * yv[9])
                  + (-9.7673534271 * yv[10]) + (-3.3944163715 * yv[11]);
        datosfiltro[k] = yv[12];
    }
    for(int i = 0; i< adq->pointsRead; i++)
        datosCanal[i] = datosfiltro[i];
```

}

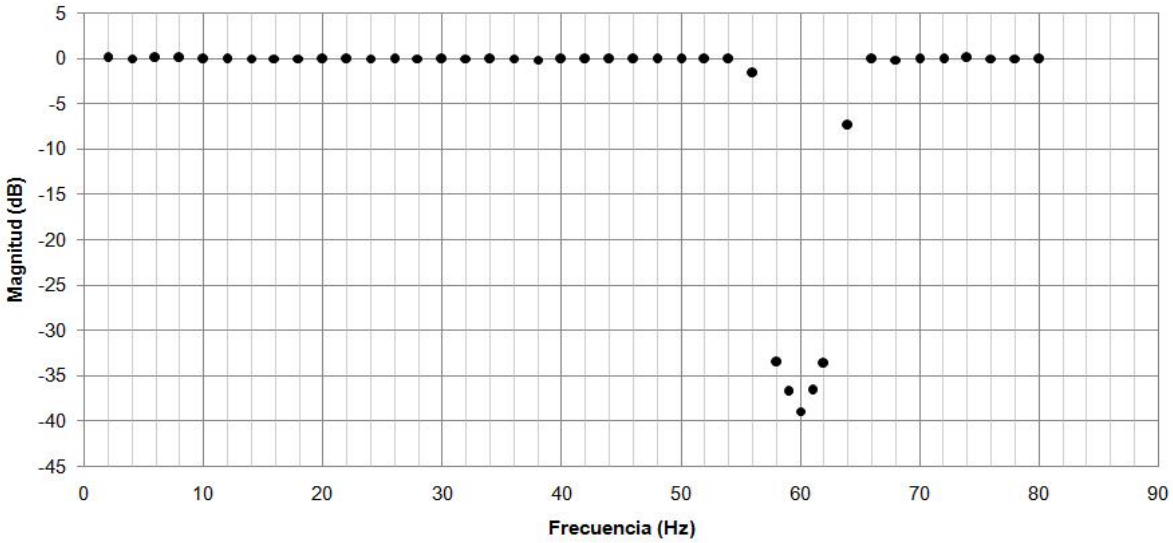
Mediante un generador de señales se introduce una señal senoidal de amplitud 1V a diferentes frecuencias, a un canal de entrada de la tarjeta para probar la eficiencia del filtro notch desarrollado, obteniéndose los valores de entrada y salida consignados en el cuadro 2.6, con estos valores es posible calcular la atenuación de la señal para cada frecuencia con la ecuación 2.5.

Cuadro 2.6. Datos de prueba del filtro notch desarrollado

Voltaje entrada(V)	**Voltaje salida(V)**	**Frecuencia(Hz)**	**Atenuación(dB)**
1.1014	1.1062	2	0.0378
0.9888	0.9884	4	-0.0035
1.0356	1.0375	6	0.0159
1.0139	1.0161	8	0.0188
0.9936	0.9846	10	-0.0790
1.0180	1.0092	12	-0.0754
1.0173	1.0176	14	0.0026
1.0014	0.9992	16	-0.0191
1.0150	1.0122	18	-0.0240
1.0072	0.9988	20	-0.0727
0.9999	0.9925	22	-0.0645
1.0039	1.0002	24	-0.0321
1.0019	0.9954	26	-0.0565
1.0000	1.0008	28	0.0069
0.9934	0.9890	30	-0.0386
1.0020	1.0019	32	-0.0009
1.0026	0.9898	34	-0.1116
1.0009	0.9986	36	-0.0200
1.0063	0.9896	38	-0.1454
0.9984	0.9934	40	-0.0436
0.9968	0.9831	42	-0.1202
0.9932	0.9871	44	-0.0535
0.9944	0.9843	46	-0.0887
0.9948	0.9908	48	-0.0350
0.9967	0.9916	50	-0.0446
1.0003	0.9909	52	-0.0820
0.9998	0.9901	54	-0.0847
0.9944	0.8233	56	-1.6401
1.0001	0.0211	58	-33.5152
0.9990	0.0145	59	-36.7639
0.9969	0.0113	60	-38.9115
0.9921	0.0146	61	-36.6441
0.9983	0.0207	62	-33.6658
0.9902	0.4240	64	-7.3671
0.9916	0.9807	66	-0.0960
0.9994	0.9831	68	-0.1428
0.9933	0.9855	70	-0.0685
1.0001	0.9935	72	-0.0575
0.9938	0.9954	74	0.0140
0.9916	0.9930	76	0.0123
0.9928	0.9929	78	0.0009
0.9929	0.9888	80	-0.0359

Se grafica frecuencia en Hertz versus la atenuación en dB para obtener la respuesta en magnitud del filtro notch como se indica en la figura 2.22.

Figura 2.22. Respuesta en magnitud del filtro notch desarrollado(dB).



Conversión de unidades de voltaje a aceleración. Como se observa en el diagrama de flujo de la figura 2.20, si el usuario desea observar los datos en unidades de aceleración (mg), a partir de los valores de voltaje leídos, se obtienen los valores de aceleración para cada uno de los canales encontrando la variable a de la ecuación 2.2, como se indica en la ecuación 2.8. Para cada eje se utilizan los valores consignados en la cuadro 2.2.

$$a = (\frac{(V_s - (V_{off} \cdot G_v))}{F_s \cdot G_v}) \cdot 1000 \tag{2.8}$$

Cuando se cumple el lapso de tiempo, se termina el ciclo y se detiene la adquisición.

Con el botón *Detener* de la ventana principal, se puede parar el proceso de adquisición y obtener el archivo con los datos leídos hasta ese momento.

Gráficas de visualización de los registros microsísmicos. La ventana principal contiene tres controles gráficos que se programan usando la biblioteca ZedGraph.dll. En cada una de las gráficas se muestran los datos adquiridos correspondientes a las señales microsísmicas de cada uno de los ejes del acelerómetro.

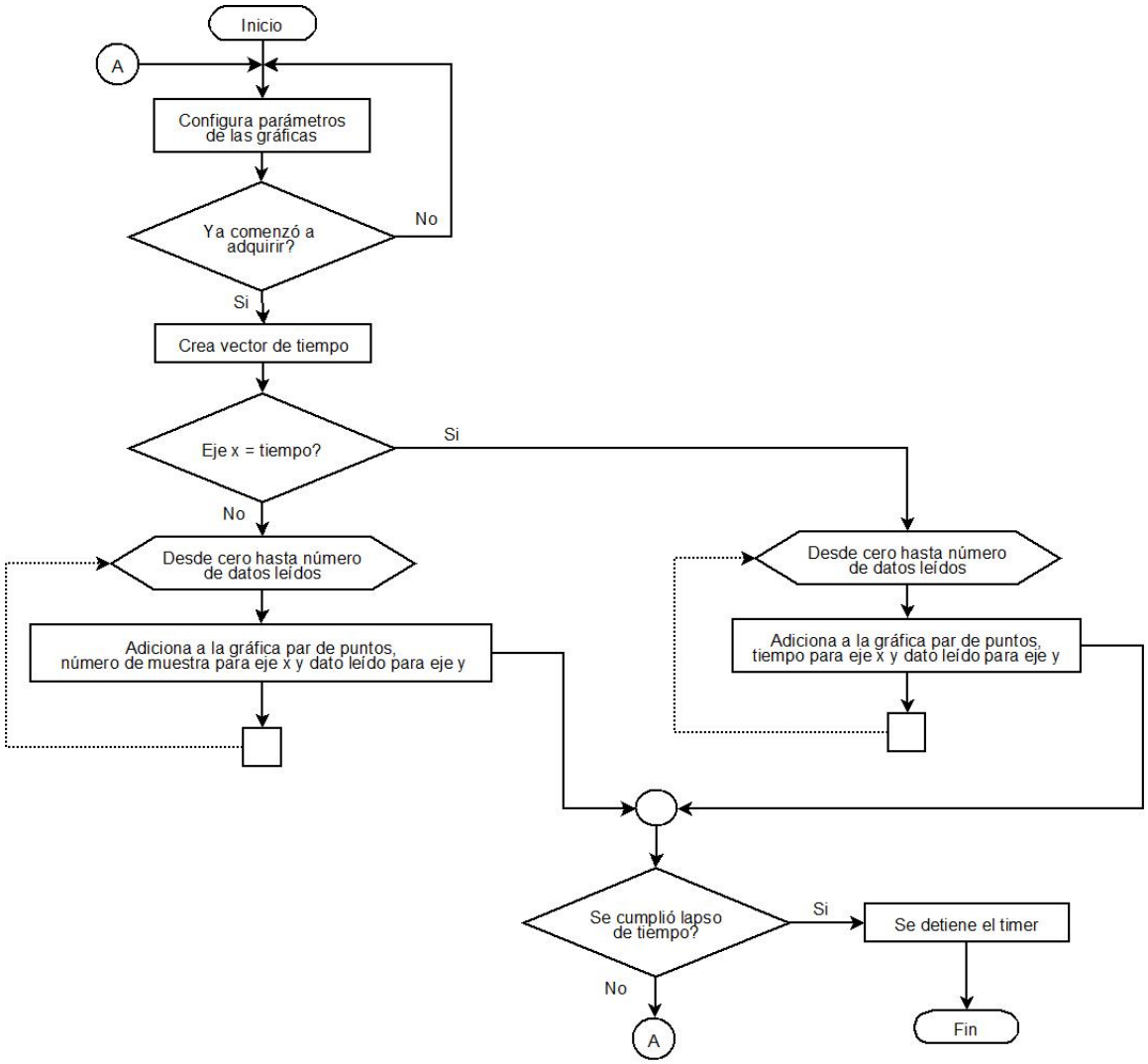
Las funciones para graficar se incluyen dentro del cuerpo de instrucciones de un *timer*, el cual lee continuamente los arrays dispuestos para contener los datos que se van a adquirir.

Al iniciar la adquisición, se comienza a graficar por intervalos de tiempo. Mediante el menú Visualizar es posible escoger las unidades del eje de las abscisas, ya sea tiempo

o número de muestras, las unidades del eje de las ordenadas, en voltaje (*voltios*) o aceleración (mg) y el canal que se desee observar.

En el diagrama de flujo de la figura 2.23 se explica la rutina implementada y el código se encuentra dentro de las instrucciones de la ventana principal en el anexo A.

Figura 2.23. Diagrama de flujo de la rutina del timer usado para graficar.



El algoritmo de programación del botón de control *Iniciar* se muestra en el diagrama de flujo de la figura 2.24 y su código completo se encuentra dentro del anexo A.

Figura 2.24. Diagrama de flujo de la rutina iniciar.

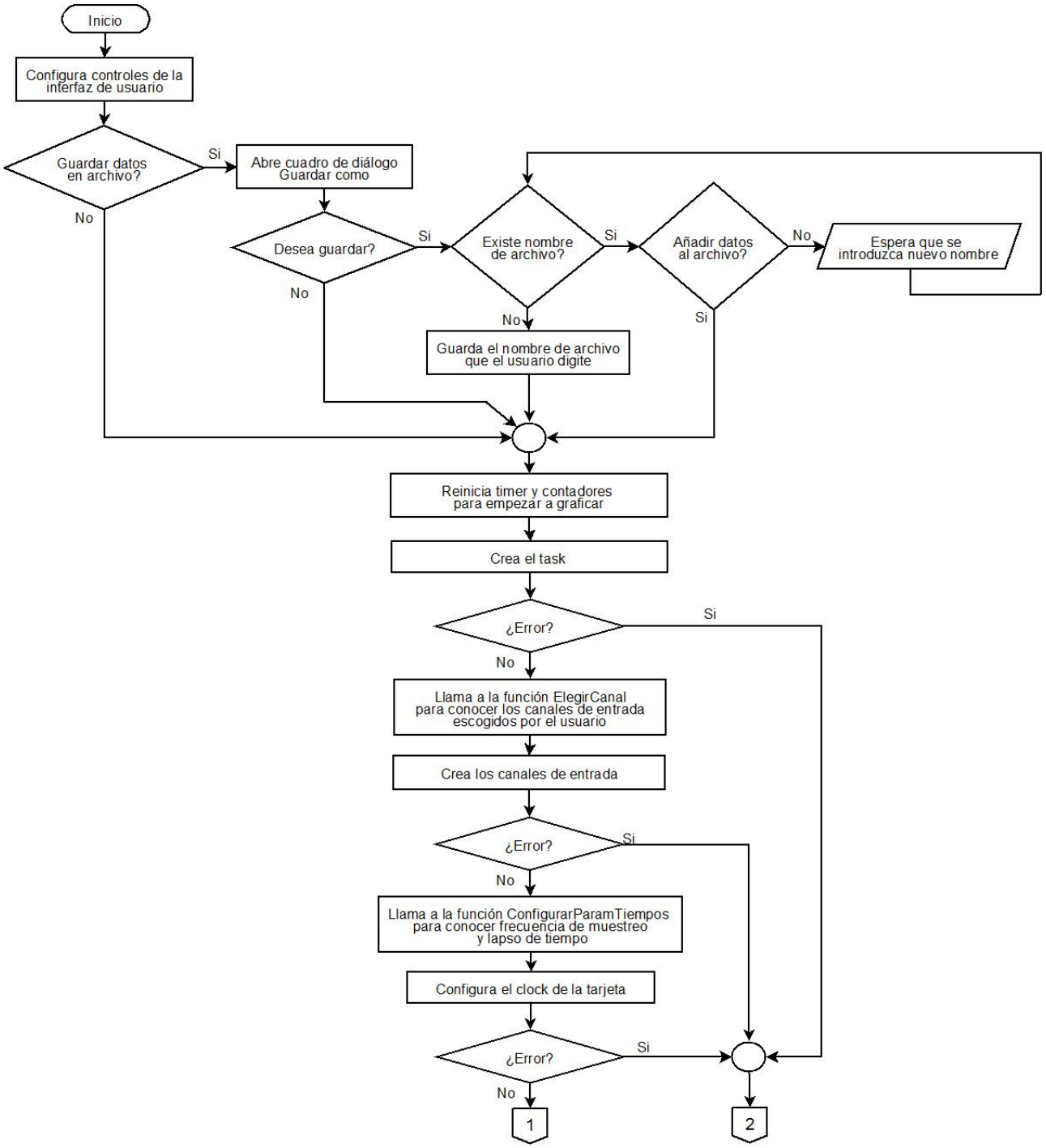
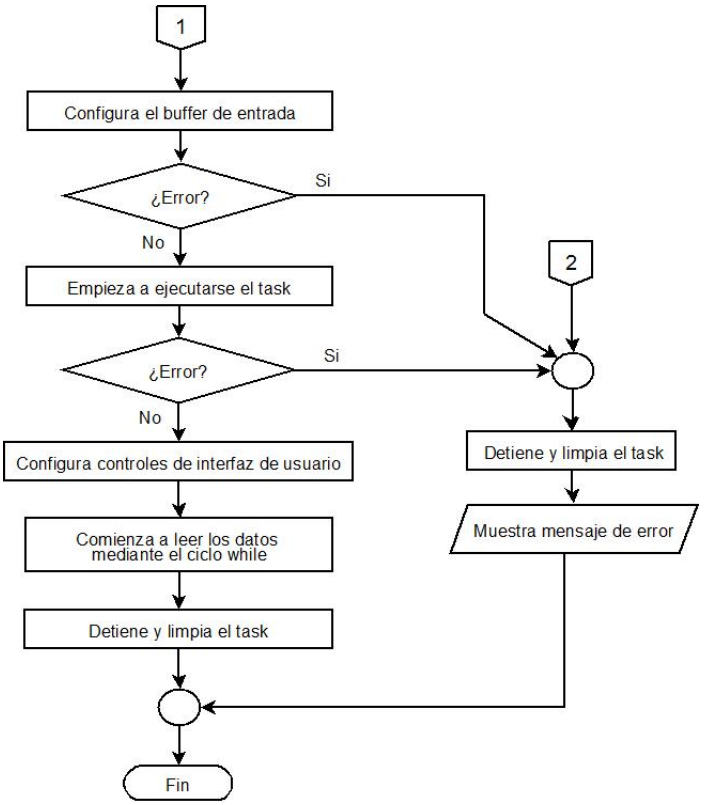


Figura 2.25. Continuación diagrama de flujo rutina iniciar.



3 ANÁLISIS Y TRATAMIENTO DE ACELEROGRAMAS

Uno de los problemas más importantes en el análisis de acelerogramas se presenta cuando se desea obtener la velocidad y el desplazamiento, debido a los siguientes factores: desconocimiento de los valores iniciales y finales de las dos variables, la longitud finita del acelerograma, corrimiento del cero del registro con respecto al cero real de aceleración y distorsiones introducidas por los sensores, el medio de registro y la forma de convertir los datos de una señal continua en el tiempo en discreta (conversión analógico-digital A/D).

Algunos otros factores, cuya influencia depende de la calidad del instrumento son: la respuesta del sensor ante aceleraciones transversales a su eje de medición, no linealidad en los circuitos de amplificación y filtrado, respuesta no lineal del propio acelerómetro en algunas frecuencias, efectos de temperatura y de humedad y envejecimiento de partes, entre otros, algunos de los cuales pueden detectarse mediante calibraciones periódicas del instrumento.

Las consecuencias en los resultados de cada uno de los factores mencionados es variable, dependiendo de las características particulares del acelerograma y del instrumento empleado para la obtención del registro.

A partir del registro obtenido se puede reproducir el movimiento del suelo o punto de la estructura donde se coloca el instrumento. Según Roca[1], para analizar acelerogramas se incluyen generalmente las siguientes partes.

- Corrección de línea de base.
- Corrección instrumental.
- Filtrado paso bajas.
- Filtrado paso altas.
- Integración numérica para la obtención de velocidades y desplazamientos.
- Análisis frecuencial.

Para realizar un análisis eficaz de los acelerogramas obtenidos en la interfaz de adquisición, se desarrolla una interfaz de usuario en el entorno Matlab para el procesamiento

[1]ROCA, ANTONI, Op. Cit, p. 147-149.

de los datos. Se utiliza Matlab debido a que es un lenguaje de alto nivel, que ofrece un ambiente interactivo para el desarrollo de algoritmos, visualización, análisis de datos y cálculo numérico[2].

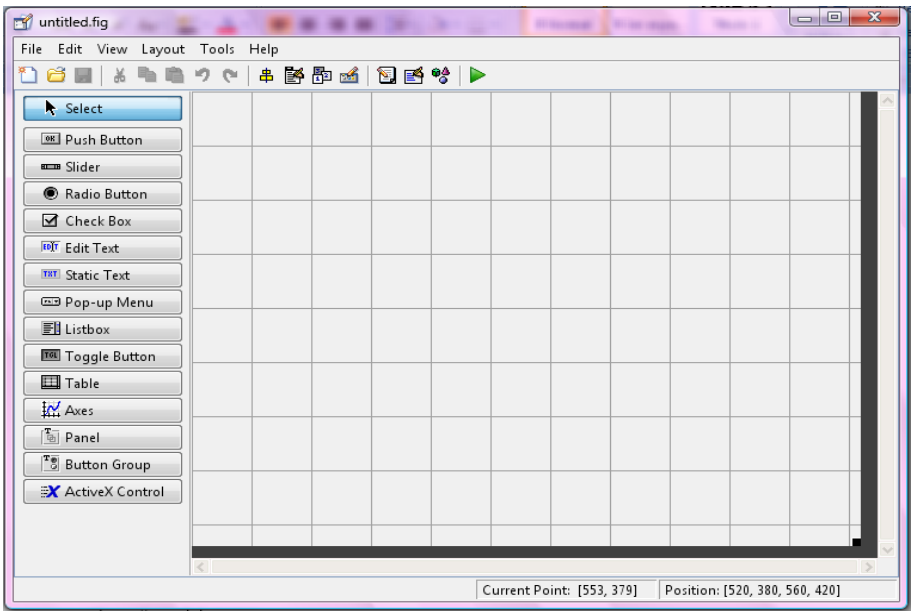
Matlab presenta un entorno versátil que puede ser usado en un amplio rango de aplicaciones y que permite integrar el código realizado con otros lenguajes de programación y otras aplicaciones, además soporta operaciones con vectores y matrices que son fundamentales para resolución de problemas en ingeniería.

3.1. DESARROLLO DE LA INTERFAZ GRÁFICA

Se desarrolla la interfaz gráfica utilizando Matlab GUIDE, el cual es una herramienta de programación visual que ofrece Matlab para poder realizar y ejecutar programas de forma simple. Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++.

Desde la ventana de comandos de Matlab se ejecuta el comando *guide* y al seleccionar *blank gui*, aparece la ventana que se muestra en la figura 3.1, donde se ubican los botones y se diseña la interfaz.

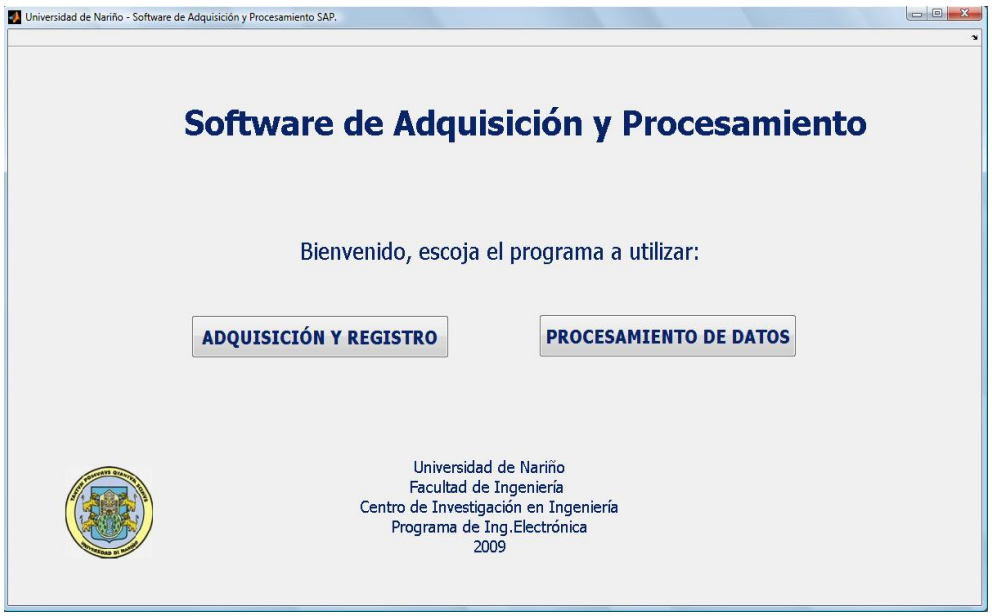
Figura 3.1. Pantalla principal de Matlab GUIDE



[2]MATHWORKS, Matlab Introduction and Key Features. Disponible en Internet: http://www.mathworks.com/products/matlab/description1.html?s_cid=ML_b1008_desintro

Se diseña la pantalla de bienvenida de la interfaz, como se indica en la figura 3.2, en donde se puede seleccionar la ejecución tanto del programa de adquisición y registro desarrollado en Visual C++, como el programa para procesamiento de datos.

Figura 3.2. Pantalla de bienvenida a la interfaz de usuario



3.2. PROGRAMA PROCESAMIENTO DE DATOS

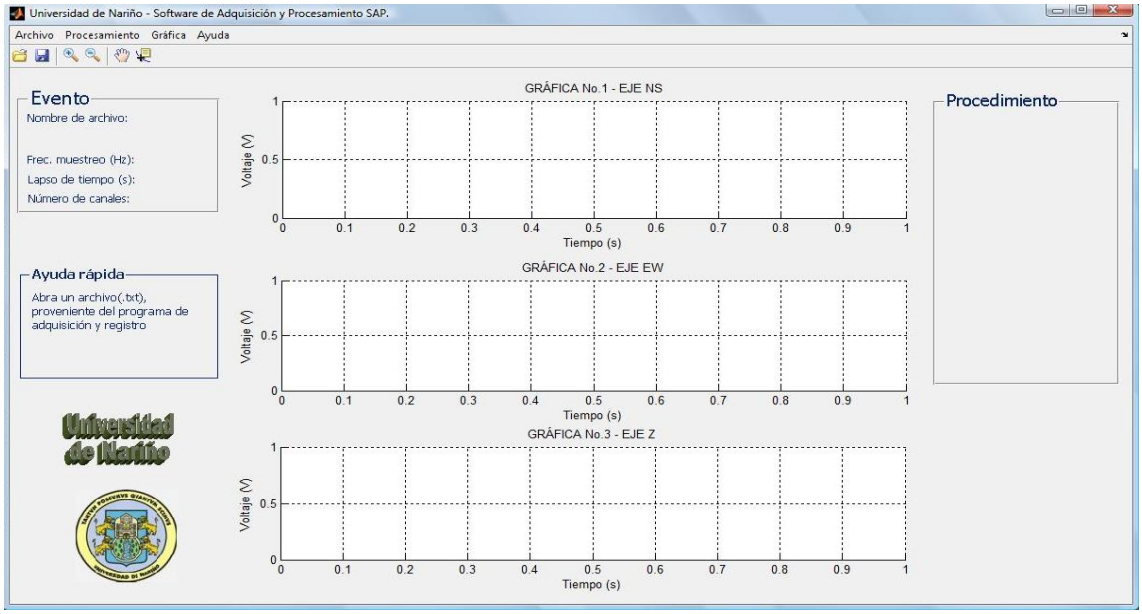
Se accede a él con un click en el botón *Procesamiento de Datos* de la ventana de bienvenida de la figura 3.2. Como se puede observar en la figura 3.3, la ventana principal del programa está compuesta por:

Barra de Título. Donde se ubica el nombre de la ventana, y los controles que permiten cerrar, minimizar o maximizar la aplicación.

Barra de Menús. Contiene cuatro menús disponibles para el usuario:

- Menú Archivo. Incluye las opciones: *Abrir*, *Guardar cálculos*, *Guardar como* y *Salir*.
- Menú Procesamiento. Contiene la opción *Corrección de Línea*, *Corrección instrumental*, *Espectros de Frecuencia*, *Integración*, *Filtros* y *Opciones* que contiene dos submenús: *Ajuste automático* y *Cambiar orden de filtro*.

Figura 3.3. Ventana principal del programa de procesamiento de datos



- Menú Gráficas. Contiene las opciones *Ejes*, *Grilla* y *Zoom*, que permiten cambiar la visualización de las gráficas.
- Menú Ayuda. Abre el archivo de ayuda del programa.

Gráficas. Existen tres gráficas, una para cada componente *z*, *ns* y *ew*, las cuales permiten visualizar los datos previamente adquiridos y su procesamiento, como en la figura 3.4.

Panel Ejes. Se visualiza al dar click en *Opciones* en la opción *Ejes* del menú *Gráficas*. Permite seleccionar la visualización de los ejes *z*, *ns* y *ew*, asi como lo ejemplifica la figura 3.5.

Panel Ayuda Rápida. Proporciona una guía rápida acerca de los procedimientos en el tratamiento de acelerogramas.

Panel Procesamiento. Informa al usuario de los pasos que ha seguido desde que inició la aplicación.

Figura 3.4. Visualización de gráficas en el panel de usuario de procesamiento de acelerogramas

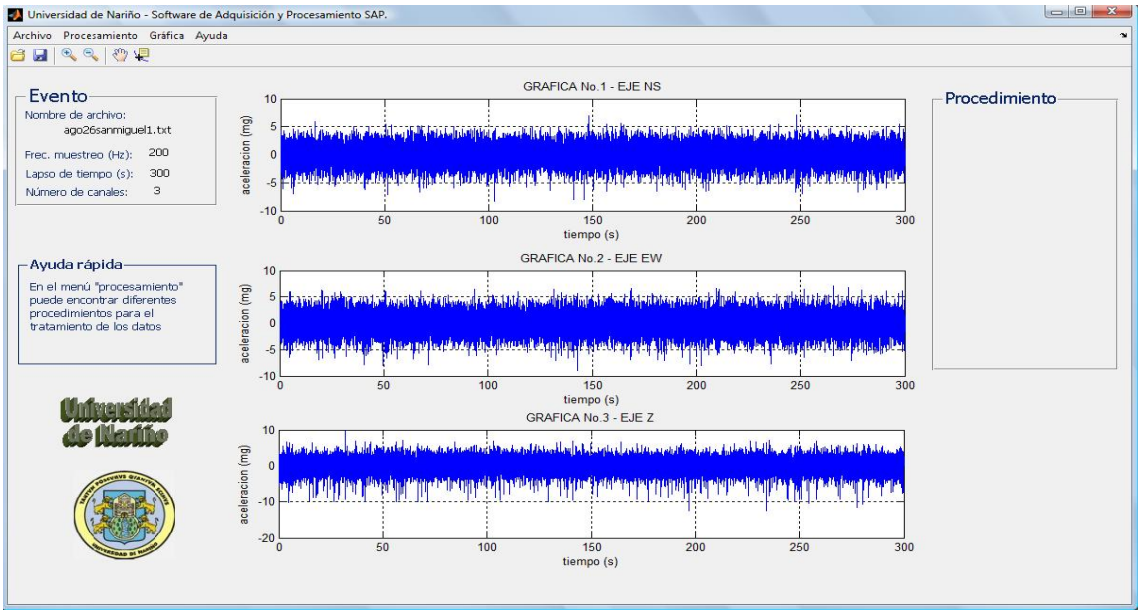


Figura 3.5. Panel de ejes

3.3. RUTINAS DE PROCESAMIENTO DE DATOS

3.3.1. Cargar archivo. Para cargar un archivo se da click en la opción *Abrir* del menú *Archivo*, esta acción llama al comando *importdata*, el cual toma la extensión del archivo y busca la función que se debe usar para abrirlo.En este caso, la extensión de los archivos a abrir es *.txt.

El archivo del registro obtenido con el programa de adquisición y registro, se compone de un encabezado cuya longitud es de 16 filas y de los datos correspondientes a los ejes x (componente horizontal Norte-Sur), y (componente horizontal Este-Oeste) y z (componente Vertical). El comando *importdata* guarda los datos en la matriz a, compuesta por dos partes: *textdata* donde guarda el encabezado y *data* donde guarda los datos numéricos.

```
a = importdata(Filename, ' ', 16);
```

Se extraen los datos correspondientes a la frecuencia de muestreo, el tiempo de adquisición, el número de canales y el número total de muestras de las líneas 9, 11, 13 y 15 de la matriz *textdata*, correspondiente al encabezado.

```
%Frecuencia de muestreo:
fm = char(a.textdata(15,1));
%Tiempo de adquisición:
dur = char(a.textdata(9,1));
%Número de canales:
ncanales = char(a.textdata(11,1));
%Número total de muestras:
totalm = char(a.textdata(13,1))
```

Definición de variables. En general, la definición de variables se hace dentro de cada rutina, exceptuando las variables que deben ser usadas a lo largo de todo el programa, como son *ch1*, *ch2* y *ch3*, correspondientes a los datos de cada uno de los canales y la variable *tiempo* que contiene los datos del tiempo de acuerdo a la frecuencia de muestreo utilizada y el número total de muestras, entre otras. Se crea el vector de tiempo con el comando *linspace*.

```
tiempo = linspace(0,dur,totalm);
```

Cuando se carga el archivo deseado en Matlab, se inicializan las variables *ch1*, *ch2* y *ch3*, de la siguiente manera:

```
ch1 = a.data(:,2);
ch2 = a.data(:,3);
ch3 = a.data(:,4);
```

Se toman las columnas dos, tres y cuatro de la matriz a para obtener los datos del canal 1 (ns), 2 (ew) y 3 (z) respectivamente.

Debido a que es necesario usar estas variables en casi todas las subrutinas, se hace uso de la herramienta *handles*, definida de la siguiente forma:

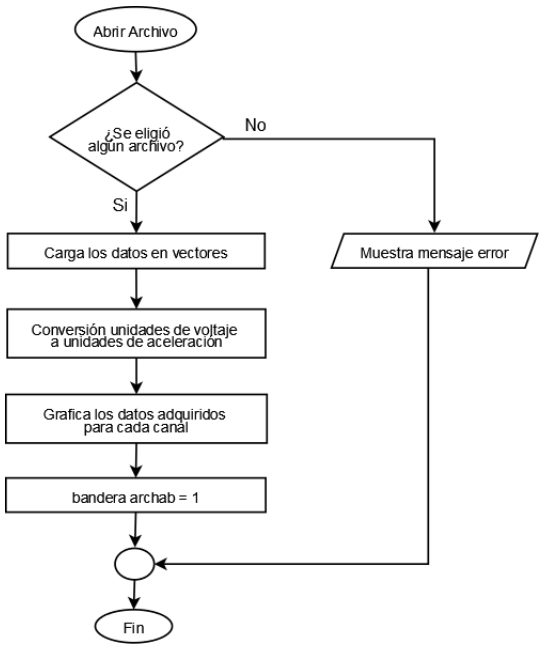
```
handles.canal1 = ch1;
handles.canal2 = ch2;
handles.canal3 = ch3;
```

Esta expresión permite colocar en *handles.canal* los valores de cada uno de los vectores, con el fin de usarlos en cualquier otro lugar del código. Se asignan los valores almacenados en *handles.canal* a nuevas variables locales, es decir, propias de cada subrutina asignándoles el mismo nombre.

```
ch1=handles.canal1;
ch2=handles.canal2;
ch3=handles.canal3;
```

El diagrama de flujo del procedimiento realizado se muestra en la figura 3.6.

Figura 3.6. Diagrama de flujo del procedimiento para cargar archivo en la interfaz de usuario de Matlab



Al cargar los datos del archivo, figura 3.7, aparece un cuadro de diálogo como el de la figura 3.8, que permite escoger la opción de realizar un ajuste automático de los datos, o de lo contrario, hacerlo manualmente.

El código para realizar esta acción es el siguiente:

```
sal = questdlg('¿Desea realizar ajuste de datos en modo automático?', 'Salir', 'Si', 'No', 'Si');

switch sal
    case 'Si'
        guidata (hObject, handles);
        correccion_Callback(hObject, eventdata, handles);
    case 'No'
        guidata (hObject, handles); %guarda los datos
end
```

En caso de que el usuario escoja el ajuste automático, la instrucción *guidata(hObject, handles)* guarda todos los cambios de las variables asignadas a funciones *handles* en la subrutina y llama a la función *instrumental_ Callback*, donde realiza la corrección instrumental de los mismos.

Figura 3.7. Ventana principal luego de cargar archivo.

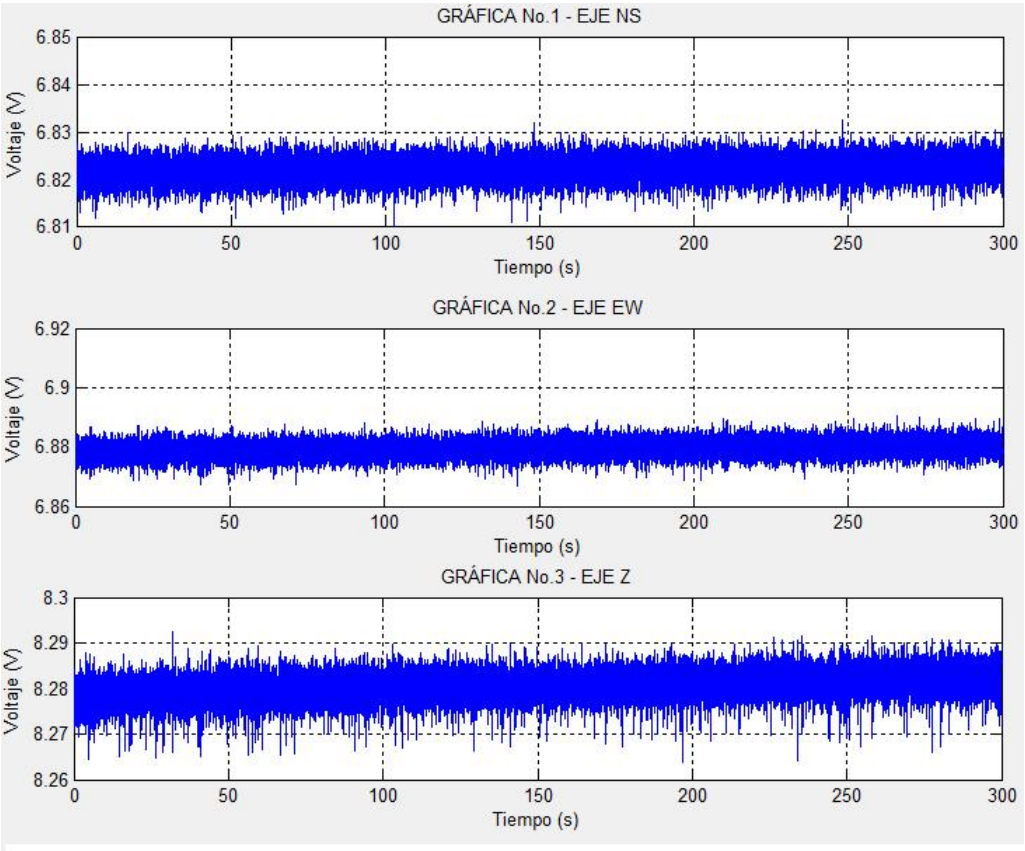
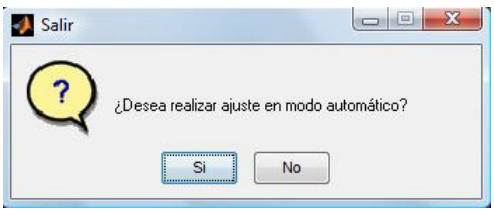


Figura 3.8. Ventana emergente para ajuste automático de datos.



3.3.2. Corrección de línea de base. A veces ocurre que los valores del acelerograma se encuentran desplazados respecto a la línea cero de aceleración, debido a varios factores como: alteraciones en el equipo, inestabilidad de la señal misma y el voltaje offset de los dispositivos utilizados.

La corrección por línea de base consiste en determinar la cantidad que se debe correr cada ordenada, para poder encontrar la línea real de cero.

Existe una gran variedad de técnicas para llevar a cabo este procedimiento, una de ellas consiste en definir una función base que puede ser una recta u otra función que se ajuste a los datos originales, en la que se aplica un procedimiento de minimización de cuadrados de las desviaciones de todos los puntos característicos del diagrama. Conviene que la función base cumpla con la condición de borde inicial (aceleración nula en $t = 0$) y que el final del registro sísmico se ajuste al acelerograma.

Rutina Corrección de línea de base. Se accede por medio de la opción *Corrección de línea* del menú *Procesamiento*. Aplica el comando *detrend(x)* a los datos de los tres canales y los grafica por medio del comando *plot*.

```
Canal 1
ch1= detrend(ch1);
axes(handles.axes1); plot(tiempo, ch1,'b');
Canal 2
ch2= detrend(ch2);
axes(handles.axes2); plot(tiempo, ch2,'b');
Canal 3
ch3= detrend(ch3);
axes(handles.axes3); plot(tiempo, ch3,'b');
```

El comando *detrend* de acuerdo a la definición en Mathworks[3], calcula un ajuste de mínimos cuadrados a partir de los datos de la señal original, a los cuales les resta la función resultante para obtener una función centrada en el orígen, eliminando el valor promedio y la tendencia lineal.

Despues de realizar la corrección de línea de base, se procede a convertir de unidades de voltaje (*voltios*) a unidades de aceleración (mg), mediante las siguientes instrucciones:

```
%conversión de voltaje a aceleración.
ch1 = (ch1/(2.7953*0.498))*1000; %para x
ch2 = (ch2/(2.8371*0.499))*1000; %para y
ch3 = (ch3/(2.7992*0.5))*1000;   %para z
```

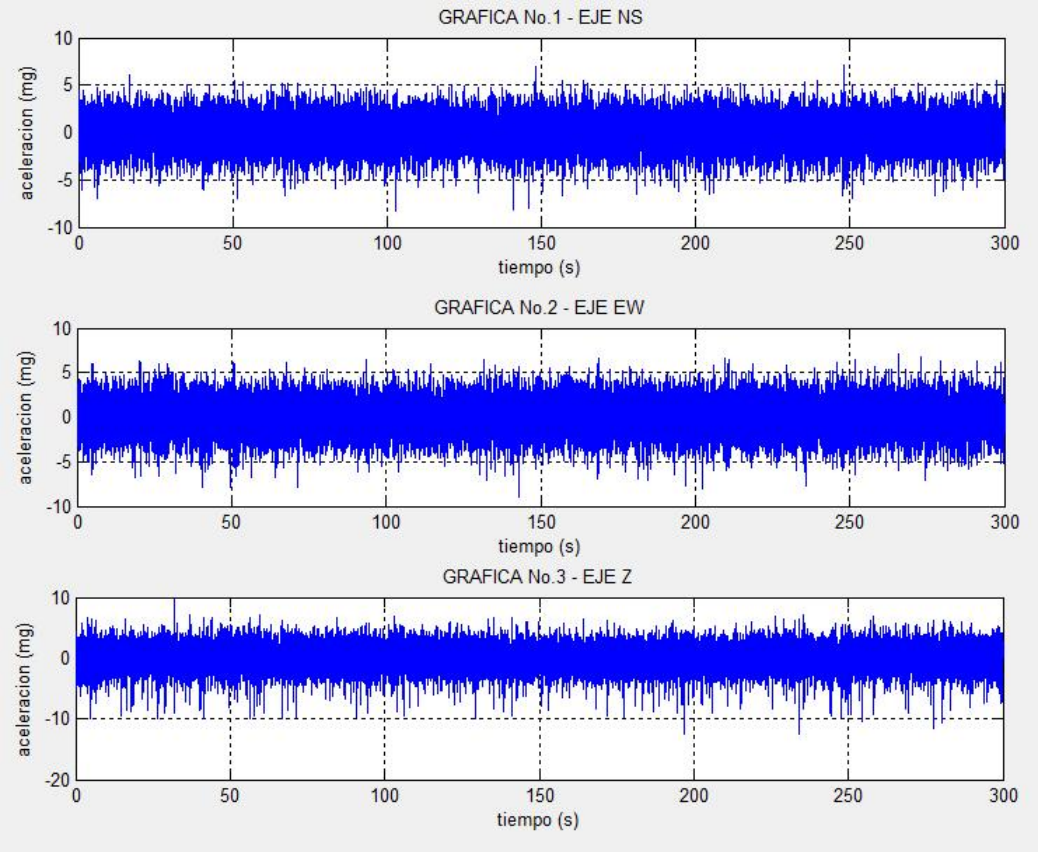
En la figura 3.9 se observan los datos de la figura 3.7, luego de aplicar la corrección de línea de base.

3.3.3. Corrección instrumental. Teniendo en cuenta los métodos de calibración del equipo detallados en el capítulo 5: Respuesta Instrumental: Pruebas y Calibración, se corrigen los datos de aceleración obtenidos.

Rutina Corrección Instrumental. Con el modelo de predicción de error (pem) de espacio de estados estimado, se obtiene la función de transferencia tanto del modelo

[3]MATHWORKS. Matlab Documentation System Identification Toolbox: detrend [en línea], 2009. Disponible en internet: http://www.mathworks.com/access/helpdesk/help/toolbox/ident/index.html?/access/helpdesk/help/toolbox/ident/ref/detrend.html&http://www.mathworks.com/support/functions/alpha_list.html?sec=2

Figura 3.9. Datos con corrección de línea de base.



dinámico del sistema, como de las posibles perturbaciones que lo puedan afectar, las cuales se toman como entradas en forma de ruido blanco gaussiano, cuya varianza es definida por el modelo de estimación para cada una de las componentes (ns, ew, z).

La rutina implementada en Matlab para realizar la corrección instrumental se compone de las siguientes sentencias:

```
function instrumental_Callback(hObject, eventdata, handles)
%Datos canal ns
ch1=handles.canal1;
%Datos canal ew
ch2=handles.canal2;
%Datos canal z
ch3=handles.canal3;

%Valores de varianza del ruido blanco gaussiano
%estimado por el modelo pem (espacio de estados)
nvz = 8.8795e-008;
nvy = 1.5085e-007;
nvx = 8.1182e-008;
```

```
%Ruido blanco gaussiano para cada eje
noisez = random('norm', 0, sqrt(nvz),size(ch3),1);
noisey = random('norm', 0, sqrt(nvy),size(ch2),1);
noisex = random('norm', 0, sqrt(nvx),size(ch1),1);

%Coeficientes función de transferencia modelo dinámico para cada eje
numz = [0  7.437e-005 -0.0002595 0.0004219  -0.0003373 0.0001141];
denz = [1  -3.988  6.866  -6.321  3.108  -0.657];
numy = [0  4.059e-005 -0.0001517  0.0002575 -0.0002049  6.348e-005];
deny = [1 -4.094  7.156  -6.626  3.236  -0.6655];
numx = [0  4.319e-005 -0.0002146 0.0004067  -0.0003589  0.0001289];
denx = [1  -4.004  6.87  -6.264   3.024  -0.62];

%Coeficientes función de transferencia modelo perturbaciones para cada eje
nnz = [1  0.4471  0.1511 0.3576 -0.1748 0.1716];
dnz = denz;
nny = [1  0.1897  -0.03695  0.3916  -0.2148 0.1605];
dny = deny;
nnx = [1  0.3467 0.07582 0.3652  -0.1712 0.1926];
dnx = denx;

%Se aplica función filter a cada eje teniendo en cuenta:
%salida = G*entrada+H*ruido
ch1 = filter(numx,denx,ch1) + filter(nnx,dnx,noisex);
ch2 = filter(numy,deny,ch2) + filter(nny,dny,noisey);
ch3 = filter(numz,denz,ch3) + filter(nnz,dnz,noisez);

guidata(hObject, handles);
end
```

Los datos corregidos se obtienen mediante la adición del efecto del modelo dinámico del sistema y del modelo de perturbaciones, sobre los datos adquiridos. En la figura 3.10 se observa la corrección instrumental aplicada a los datos de la figura 3.7.

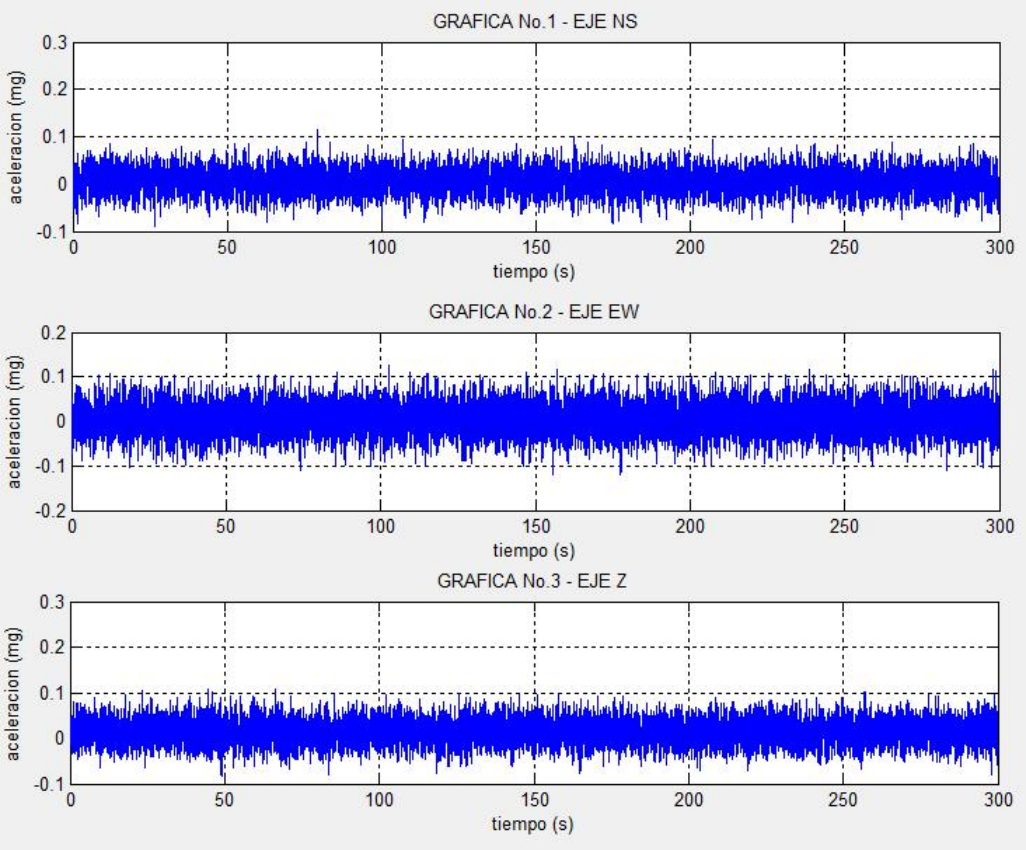
3.3.4. Filtrado. Un filtro es un elemento que discrimina una determinada frecuencia o gama de frecuencias de una señal que pasa a través de él, pudiendo modificar tanto su amplitud como su fase.

Características. Las características que definen un filtro vienen determinadas por los siguientes conceptos:

- Función de transferencia. Describe la forma de comportarse un filtro, determina la forma en que la señal aplicada cambia en amplitud y en fase al atravesarlo. La función de transferencia elegida tipifíca el filtro.
- Orden. Describe el grado de aceptación o rechazo de frecuencias por arriba o por debajo de la respectiva frecuencia de corte, que es el punto a partir del cual el filtro empieza a recortar frecuencias.

Tipos. Hay dos tipos principales de filtros: analógico y digital.

Figura 3.10. Datos con corrección instrumental aplicada.



- Un filtro analógico emplea circuitos electrónicos con componentes discretos como: resistencias, condensadores y amplificadores operacionales. Tales filtros son muy empleados para reducción de ruido, mejoramiento de señales, ecualizadores gráficos y muchas otras áreas.

- Un filtro digital emplea un procesador digital que efectúa operaciones matemáticas en los valores muestreados de una señal. El procesador puede ser de propósito general, tal como un computador personal, un chip DSP (Procesador Digital de Señales) especializado o una FPGA programable.

Las ventajas de usar filtros digitales sobre los analógicos son principalmente las siguientes:

- Un filtro digital es programable, es decir, su funcionamiento está determinado por un programa almacenado en la memoria contigua al procesador; esto significa

que puede ser variado fácilmente sin afectar al hardware, mientras que la única manera de variar un filtro analógico es alterando el circuito.

- Los filtros digitales pueden ser fácilmente diseñados, probados e implementados en un computador. Los analógicos pueden ser simulados, pero siempre hay que implementarlos a través de componentes discretos para ver su funcionamiento real.
- Las características de los filtros analógicos, particularmente los que contienen componentes activos, están sujetos a alteraciones y dependen de la temperatura. Los filtros digitales no sufren estos problemas y son extremadamente estables ante factores externos.
- A diferencia de los filtros analógicos, los digitales pueden manejar las bajas frecuencias con mucha precisión.
- Los filtros digitales son mucho más versátiles a la hora de manipular la señal, puesto que pueden llegar a variarla y tratarla radicalmente cambiando sus características.

Vallverdú, Rodríguez y Moreno[4], clasifican a los filtros en **recursivos** y **no recursivos**. Un filtro *no recursivo* es aquel cuya salida está calculada exclusivamente a partir de valores de entrada ($Yn = Xn + Xn-1 + Xn-2...$), mientras que uno *recursivo* es aquel que además de los valores de entrada emplea valores previos de salida ($Yn-1, Yn-2...$), los cuales se almacenan en la memoria del procesador. La palabra recursivo significa literalmente "volver hacia atrás" y se refiere al hecho de que valores de salida previamente calculados son usados para calcular los nuevos valores de salida.

Puede parecer que los filtros recursivos requieren más cálculos para ser ejecutados, pero en realidad, un filtro recursivo generalmente requiere menos coeficientes para ser evaluado, es decir, que es de menor orden y más corto que un filtro no recursivo.

Los filtros no recursivos se conocen como *filtros FIR* (Respuesta al Impulso Finita) y los recursivos como *filtros IIR* (Respuesta al Impulso Infinita). Estos términos se refieren a las diferentes respuestas al impulso de ambos tipos de filtros. La respuesta al impulso de un filtro digital es la secuencia de salida cuando se aplica un impulso unidad a su entrada.

Filtro FIR. En este tipo de filtro digital si su entrada es un impulso, la salida será un número limitado de términos no nulos. Su expresión en el dominio discreto es:

$$y_n = \sum_{k=0}^{N-1} b_k x(n-k) \tag{3.1}$$

[4]VALLVERDÚ, Francesc, RODRÍGUEZ, José A. y MORENO, Asunción. Tratamiento digital de la señal - Una introducción experimental. Barcelona : s.n., 1995. p. 213-252

El orden del filtro está dado por el número de coeficientes N. También la salida puede ser expresada como la convolución de una señal de entrada $x[n]$ con un filtro $h[n]$:

$$y_n = \sum_{k=0}^{N-1} h_k x_n - k \tag{3.2}$$

Los filtros FIR son estables puesto que sólo tienen polos, es decir, elementos en el numerador en su función de transferencia. También tienen la ventaja que pueden diseñarse para ser de fase lineal, por lo cual, no introducen desfases en la señal, a diferencia de los filtros IIR o los filtros analógicos.

Sin embargo, tienen el inconveniente de ser más largos al tener más coeficientes que los IIR, capaces de cumplir similares características, esto requiere un mayor tiempo de cálculo que puede dar problemas en aplicaciones en tiempo real.

Filtro IIR. Es un tipo de filtro digital que si su entrada es un impulso, la salida será un número ilimitado de términos no nulos, lo cual significa que nunca volverá a un estado de reposo.

Para obtener la salida se emplean valores de entrada actuales y anteriores, además, valores de salida anteriores que son almacenados en memoria y realimentados a la entrada. Su expresión en el dominio discreto es:

$$y[n] = \sum_{i=0}^{P} b_i x[n-i] - \sum_{j=1}^{Q} a_j y[n-j] \tag{3.3}$$

El orden del filtro está dado por el máximo entre P y Q.

Las ventajas de los filtros IIR respecto a los FIR es que pueden conseguir una misma respuesta empleando un número menor de coeficientes, requiriendo un menor tiempo de cálculo. El inconveniente es, además de su inestabilidad, la introducción de desfases en la señal, que pueden ser compensados añadiendo más coeficientes al filtro.

Filtros ideales. Según Coral[5], los filtros ideales son aquellos que están definidos como una función escalón, del siguiente modo:

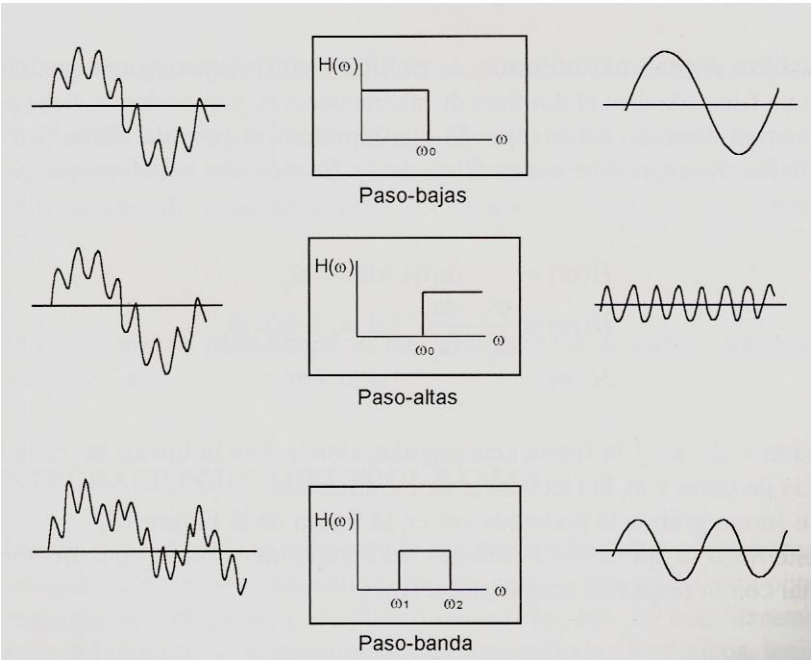
- Filtro ideal pasa bajas: elimina todas las frecuencias superiores a una dada ω_0, y deja pasar sin atenuación las frecuencias inferiores.
- Filtro ideal pasa altas: elimina todas las frecuencias inferiores a una dada ω_0, y deja pasar sin atenuación las frecuencias superiores.

[5]CORAL MONCAYO, Hugo. Ingeniería Sismológica. Pasto: Universidad de Nariño. Departamento de Ingeniería Civil (2009) p. 114.

- Filtro ideal pasa banda: elimina todas las frecuencias inferiores a una dada ω_1, y las superiores a ω_2, y deja pasar sin atenuación las comprendidas entre ω_1 y ω_2.

Un ejemplo gráfico de estos filtros se muestra representado en la figura 3.11, donde $H(\omega)$ es una función filtro.

Figura 3.11. Comportamiento de los filtros ideales: pasa bajas, pasa altas y pasa banda.



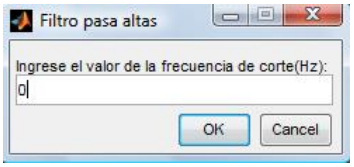
Fuente: CORAL MONCAYO, Hugo. Ingeniería Sismológica. Pasto: Universidad de Nariño. Departamento de Ingeniería Civil (2009) p. 115.

Rutina aplicar filtro. Los filtros a usar son de tipo IIR. Para aplicar un filtro se escoge la opción *Filtros* del menú *Procesamiento*, éste despliega el panel filtros que se observa en la figura 3.12, el cual permite seleccionar el tipo de filtro que se desea usar, puede ser: pasa altas, pasa bajas o pasa banda.

Figura 3.12. Panel de filtros

Luego aparece una ventana emergente como la de la figura 3.13, donde se introduce el rango de las frecuencias a filtrar.

Figura 3.13. Ventana emergente para seleccionar la frecuencia del filtro a usar



El filtro que se aplica es por defecto un filtro Butterworth de orden dieciseis. Se escoge este tipo porque es el de mejor respuesta para el rango de frecuencias de interés, además la aproximación de Butterworth es la que presenta una fase más próxima a la ideal para un orden dado, pero el orden que necesita para cumplir las especificaciones suele ser notablemente mayor al que requieren los demás tipos de filtro.

Para elegir el orden del filtro se utiliza la función *buttord* de Matlab. En la documentación de Matlab[6] se precisa que esta función retorna el orden de filtro que no pierde mas de Rp dB en la banda de paso y que tiene una atenuación de al menos Rs dB en la banda a rechazar, además tiene en cuenta las frecuencias de la banda de transición Wp y Ws, de la banda de paso y rechaza banda respectivamente, normalizadas desde 0 hasta 1, donde 1 corresponde a la mitad de la frecuencia de muestreo, es decir, $\frac{F_s}{2} = 100Hz$.

Por ejemplo si se desea un filtro Butterworth con valores de atenuación $Wp = 50Hz$, $Ws = 66Hz$, $Rp = 3dB$ y $Rs = 70dB$, se utilizan las siguientes instrucciones en Matlab:

```
Wp = 0.5;  %50Hz normalizados
Ws = 0.66; %66Hz normalizados
Rp = 3;  %dB
Rs = 70;  %dB

[N, Wn] = buttord(Wp,Ws,Rp,Rs); %N:orden, Wn:frec. de corte
```

Al aplicar los datos se obtiene como resultado un filtro de orden dieciseis con frecuencia de corte en 49.5Hz.

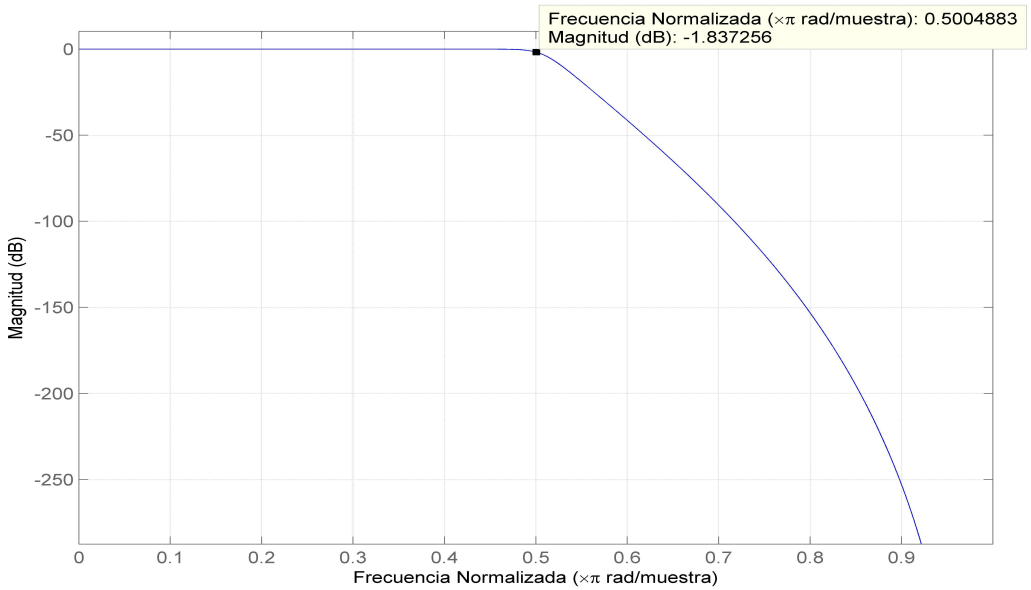
Por medio de la función *butter* en los vectores A y B, se encuentran los coeficientes de la función de transferencia que caracteriza al filtro, donde B corresponde al numerador y A al denominador. Se toman como argumentos de entrada los valores de N y Wn encontrados en el procedimiento anterior. Dependiendo del rango de frecuencias a filtrar a partir de la frecuencia de corte, es posible obtener un filtro pasa bajas, pasa altas o pasa banda con las siguientes sentencias:

```
[B,A] = butter(N,Wn, 'low');   %para filtro pasa bajas
[B,A] = butter(N,Wn, 'high');  %para filtro pasa altas
[B,A] = butter(N,Wn);   %donde Wn = [Wn1 Wn2] para filtro pasa banda
```

[6]Matlab Help Browser. Doc buttord.

Usando la herramienta *fvtool*, la cual calcula la respuesta en magnitud de un filtro, se visualizan las respuestas de los filtros diseñados anteriormente. En las figuras 3.14, 3.15 y 3.16, se observa el comportamiento del filtro pasa bajas, pasa altas y pasa banda respectivamente.

Figura 3.14. Respuesta filtro pasa bajas



Para aplicar los filtros a los datos se utiliza el comando *filtfilt*, el cual toma los vectores B y A calculados con la rutina anterior y procesa los datos de entrada.

Luego estos datos se grafican de manera automática en la ventana principal y el usuario puede escoger si desea guardar los cambios realizados dando un click en el botón *OK* que aparece en la ventana de filtros o deshacerlos, dando un click en el botón *Cancelar*.

Como ejemplo se aplica a los datos de la figura 3.10, un filtro pasa bajas con frecuencia de corte en 10Hz, su resultado se observa en la figura 3.17.

El diagrama de flujo del procedimiento que se lleva a cabo en la rutina filtros se muestra en la figura 3.18.

Figura 3.15. Respuesta filtro pasa altas

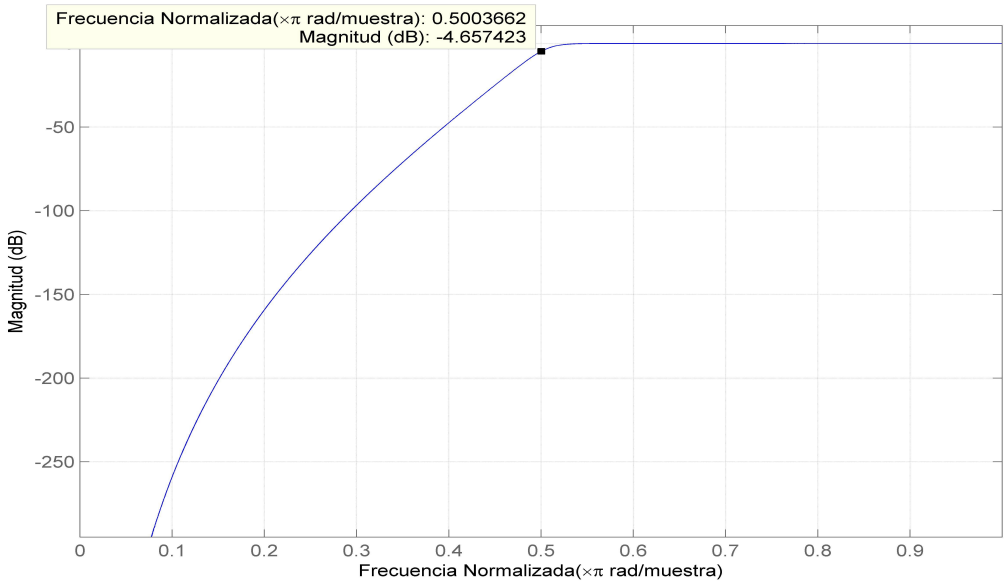
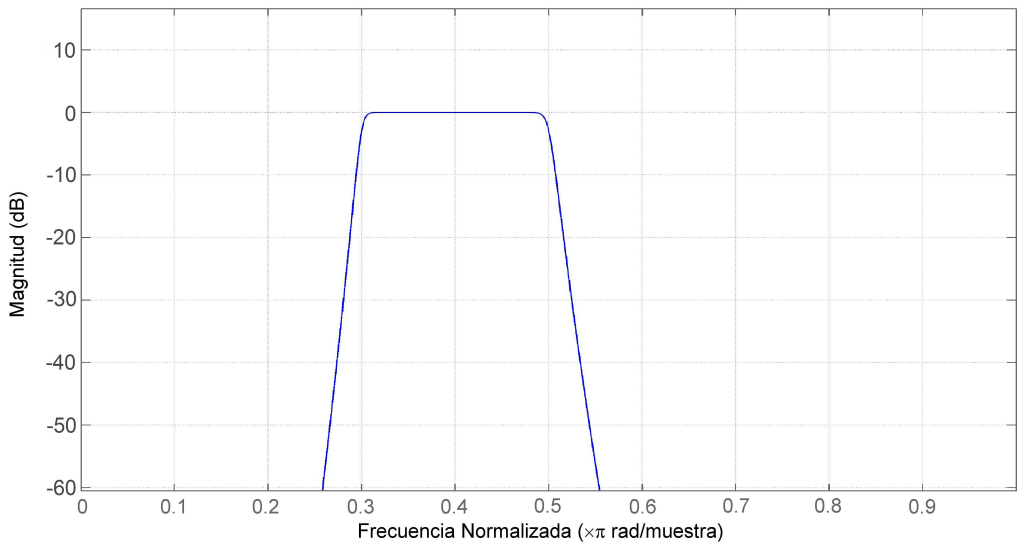
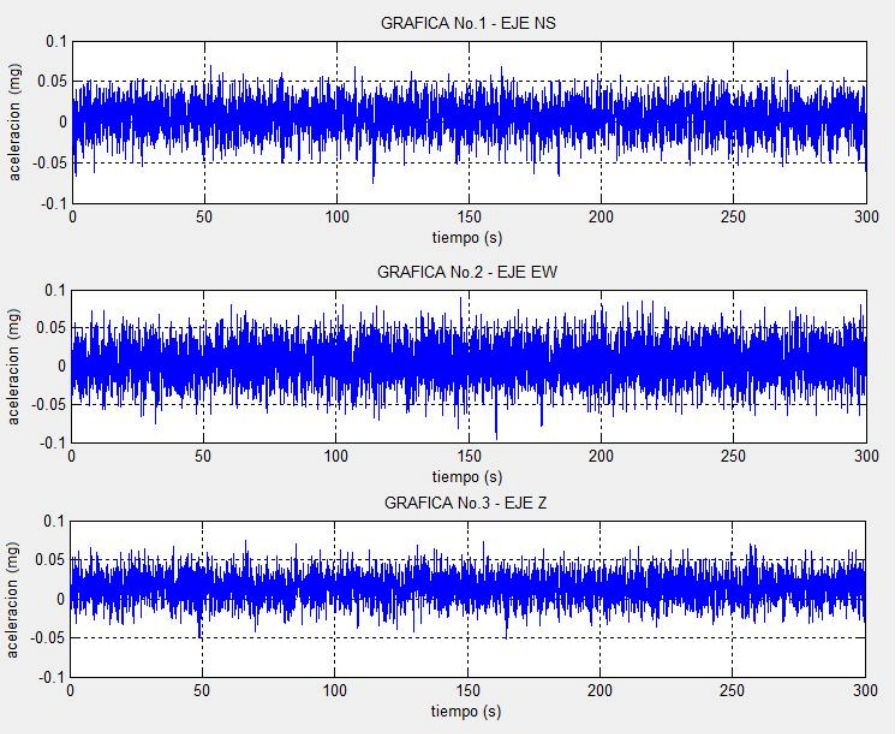


Figura 3.16. Respuesta filtro pasa banda



3.3.5. Integración numérica para la obtención de velocidades y desplazamientos. Cuando el registro temporal de aceleración ha sido convenientemente co-

Figura 3.17. Datos con filtro pasabajas con frecuencia de corte de 10Hz aplicado.



rregido para su posterior utilización, se puede obtener los registros de velocidad y desplazamiento a través de la integración.

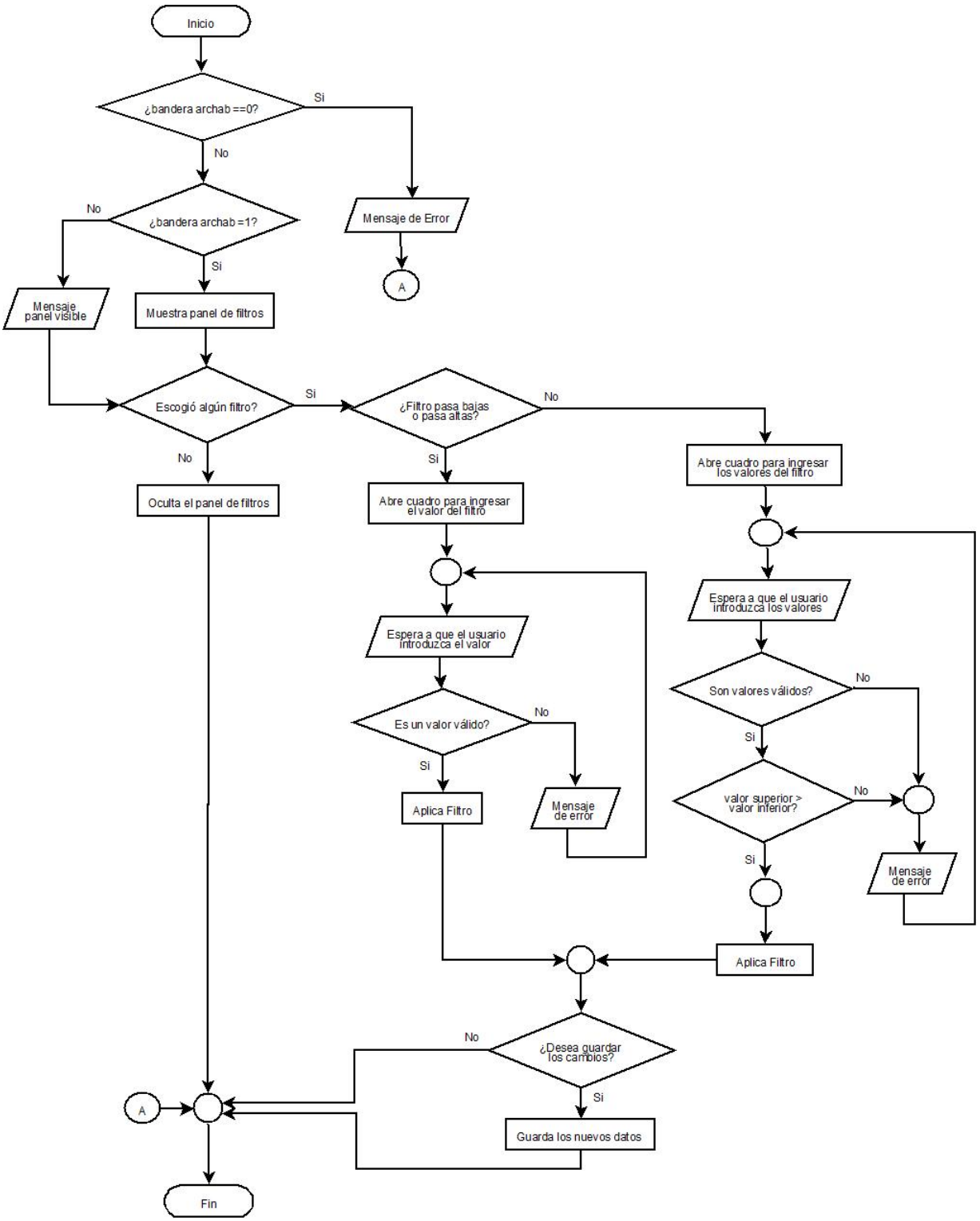
Si se considera el dominio del tiempo, la integración se hace por métodos numéricos, teniendo en cuenta las ecuaciones 3.4 y 3.5

$$\vec{v}(t) = \int_{t_0}^{t} \vec{a}(t) \cdot dt + \vec{v}_0 \tag{3.4}$$

$$\vec{d}(t) = \int_{t_0}^{t} \vec{v}(t) \cdot dt + \vec{d}_0 \tag{3.5}$$

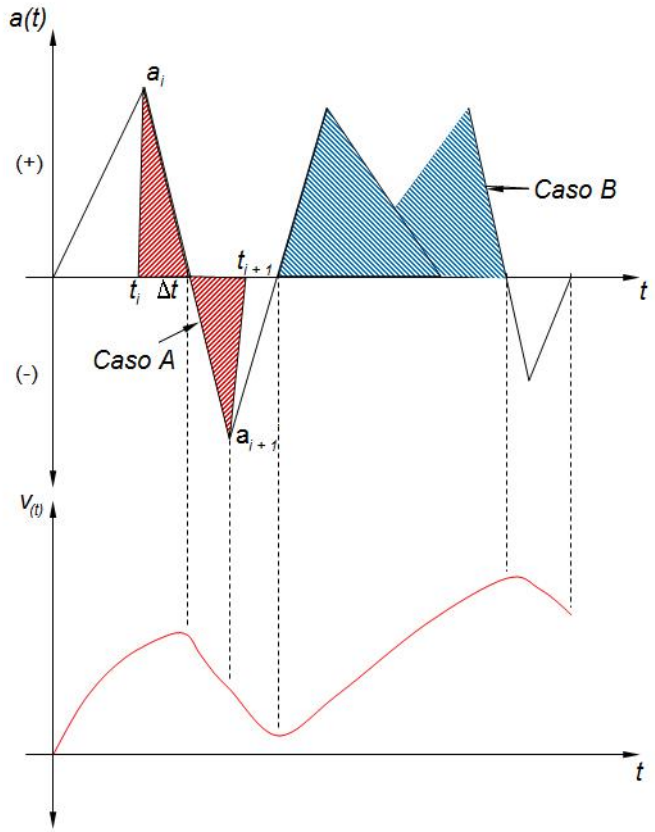
Para calcular el área bajo una curva se puede utilizar la integral definida como se describió anteriormente, pero en esta ocasión no es posible calcular con exactitud el área utilizando este recurso, ya que no se conoce una expresión analítica para la función primitiva, sino que el integrando está definido por una tabla de valores.

Figura 3.18. Diagrama de flujo del procedimiento para aplicar filtros.



Por lo tanto, se realiza una estimación o una aproximación del valor del área. Coral[7] propone dos casos en el cálculo del área, como se observa en la figura 3.19.

Figura 3.19. Casos para la integración a partir de la aceleración. Parte baja: obtención del velocigrama a partir del acelerograma, realizando la acumulación de las áreas con su correspondiente signo.



Fuente: CORAL MONCAYO. Op. cit., p. 379.

Caso A: cuando la curva de aceleración o velocidad corta al eje del tiempo; se tiene por semejanza de triángulos:

$$\frac{\Delta t}{a_i} = \frac{t_{i+1} - t_i}{a_i + a_{i+1}} \rightarrow \Delta t = \frac{|a_i| \cdot |t_{i+1} - t_i|}{|a_i| + |a_{i+1}|} \tag{3.6}$$

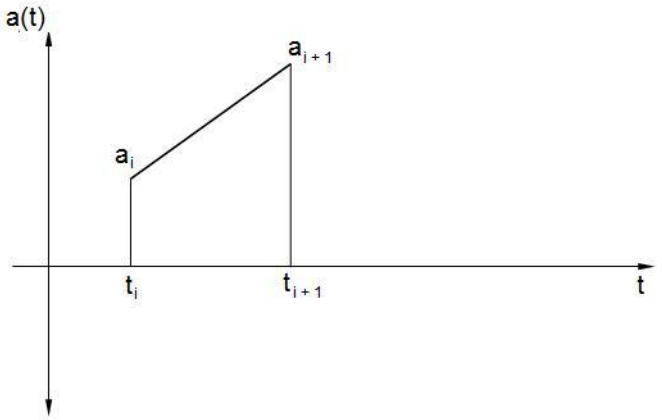
El área bajo la curva entre t_i y t_{i+1} es:

$$A_i = \frac{a_i \cdot \Delta t}{2} + \frac{a_{i+1}}{2}(t_{i+1} - t_i - \Delta t) \tag{3.7}$$

[7]CORAL MONCAYO. Op. cit., p. 378-380.

Caso B: la curva de aceleración o velocidad no corta al eje del tiempo; el área bajo la curva es la de un trapecio como en la figura 3.20.

Figura 3.20. Obtención del área cuando se trata del caso B de integración.



Fuente: CORAL MONCAYO. Op. cit., p. 380.

Matemáticamente, el área del trapecio esta descrita por:

$$A_i = \frac{1}{2} \cdot (a_i + a_{i+1}) \cdot (t_{i+1} - t_i) \tag{3.8}$$

Se debe tener especial cuidado de los signos.

Conforme lo anterior, se realiza la integración asumiendo que las líneas que describen al acelerograma son rectas y se reduce el proceso a acumular las áreas de los triángulos y trapecios que se conforman. La primera condición de borde para la integración numérica de la aceleración es que para un tiempo "cero", la velocidad debe ser "cero".

El diagrama que se obtiene luego de la integración, presenta errores numéricos como producto de suponer que el acelerograma está compuesto por una secuencia de líneas rectas. Existe una gran variedad de técnicas que permiten mejorar los diagramas obtenidos, en esta ocasión, la técnica utilizada es la propuesta por Romo Proaño[8], la cual consiste en aplicar un proceso de minimización de cuadrados de las desviaciones de todos los puntos del diagrama. Conviene igualmente que cumpla también con la condición de borde inicial "cero".

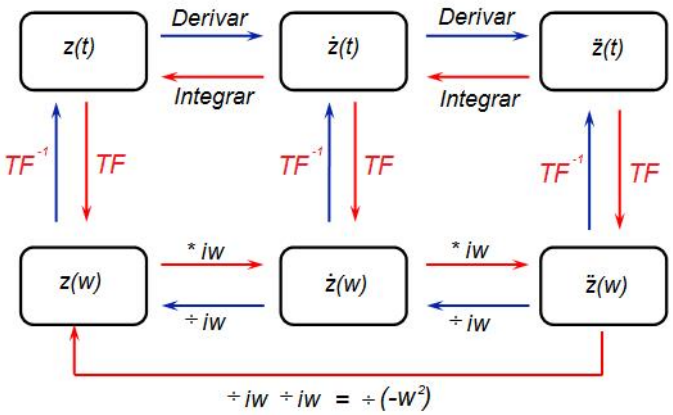
[8]ROMO PROAÑO, Marcelo. Técnica para la generación de diagramas de velocidades y desplazamientos a partir de acelerogramas sísmicos. Quito : Centro de Investigaciones científicas, Escuela politécnica del ejército, 2003. p. 4,5,8.

Cada integración suaviza los componentes más altos del movimiento de las ondas, por lo que los registros de velocidad y desplazamiento son más simples y permiten una interpolación directa de los patrones de las ondas. Donde exista un pico de máxima velocidad es una clara medida del contenido de frecuencias intermedias de un sismo.

Los picos de desplazamiento muestran las características de largo período del movimiento; por tanto, con los tres valores pico (aceleración, velocidad y desplazamiento) se tiene una importante descripción del movimiento del suelo, mostrando cada uno, una región diferente del espectro de frecuencias.

En el diagrama de la figura 3.21, se muestra el procedimiento general para la obtención de velocidad, desplazamiento y aceleración, tanto en el dominio temporal como en el frecuencial.

Figura 3.21. Obtención aceleración, velocidad y desplazamiento a partir del acelerograma. Campo temporal: desplazamiento $z(t)$, velocidad $\dot{z}(t)$ y aceleración $\ddot{z}(t)$. Campo frecuencial: desplazamiento $z(\omega)$, velocidad $\dot{z}(\omega)$ y aceleración $\ddot{z}(\omega)$. TF: transformada de Fourier. TF^{-1}: Transformada inversa de Fourier.



Fuente: CORAL MONCAYO. Op. cit., p. 116.

Rutina integración. Se define la variable *integ*, utilizada como bandera, con el fin de controlar la aplicación del procedimiento. Esta bandera se inicializa en "cero" al ejecutar el programa y toma su valor actual de la función *handles.inte*. En caso de que se ejecute por primera vez la rutina de integración, es decir, para obtener el diagrama de velocidades, *integ* toma el valor "uno". Si se ejecuta por segunda vez para obtener el diagrama de desplazamientos, *integ* toma el valor "dos", de manera que ya no es posible realizar nuevamente el procedimiento.

La corrección de línea de base, es un requisito para realizar el proceso de integración, por tanto, se utiliza de igual forma la bandera *corr*, la cual no permite realizar el

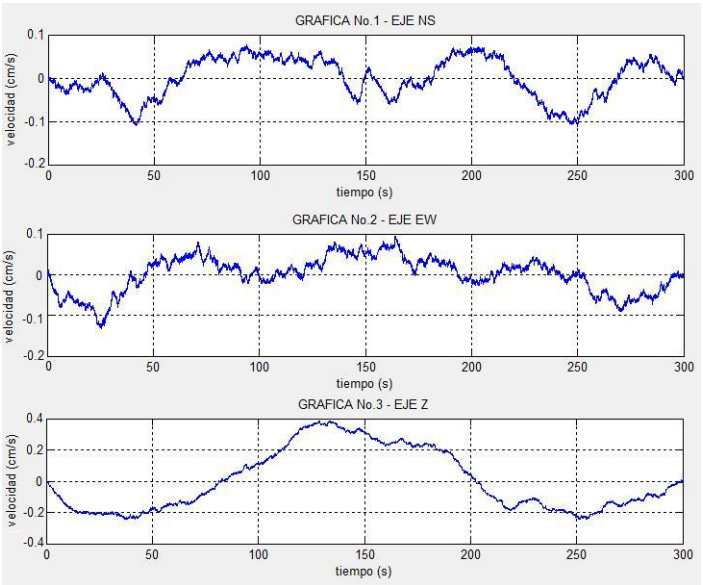
procedimiento, si no se aplica primero la corrección de línea de base y muestra un mensaje de error.

El código en Matlab para realizar la integración del acelerograma es el siguiente:

```
integ=handles.inte;
corr=handles.cor;

if corr==0
   errordlg('Corrección de linea de base no aplicada', 'Error');
elseif integ<2
   ...realiza procedimiento, calculando área de triángulo o trapecio
   dependiendo del caso.
else
   errordlg('No es posible realizar procedimiento otra vez.');
end
```

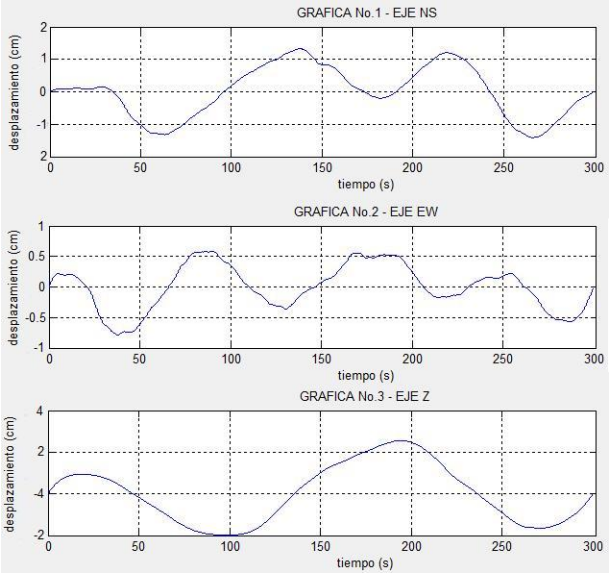
Al aplicar la rutina de integración sobre los datos de la figura 3.10 por primera vez, se obtiene el diagrama de velocidades de la figura 3.22.

Figura 3.22. Ejemplo de diagrama de velocidades.



Sobre estos datos obtenidos se aplica nuevamente el proceso de integración para obtener el diagrama de desplazamiento que se observa en la figura 3.23.

Figura 3.23. Ejemplo de diagrama de desplazamiento.



3.3.6. Análisis frecuencial. Para realizar un análisis del contenido en frecuencias del acelerograma, se utiliza el espectro de Fourier.

El **espectro de Fourier** es un parámetro que proporciona un amplio conocimiento acerca del contenido en frecuencias del acelerograma, de modo que para cada período de oscilación del acelerograma se representa la amplitud máxima que le corresponde.

De la interpretación del espectro de Fourier, se deducen los períodos más importantes de un microsismo, en los lugares donde se alcanzan los picos de amplitud. Además, es posible encontrar la frecuencia o período predominante de un registro, que es aquel para el cual el espectro de Fourier de la aceleración alcanza su valor máximo y que además mantiene una correlación directa con la distancia epicentral.

Se suele representar con ejes logarítmicos, de modo que en el eje de las abscisas se colocan las distintas frecuencias de oscilación y en el eje de las ordenadas se representan las amplitudes.

Para realizar un análisis espectral de la señal es necesario su paso del dominio del tiempo al dominio de la frecuencia, es decir, la descomposición de la señal en productos de senos y cosenos, esto es posible mediante la transformada de Fourier descrita matemáticamente en la ecuación 3.9.

$$X(\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega t} dt \tag{3.9}$$

Donde X(ω) es la transformada de Fourier de *x(t)*.

Esta transformada hace referencia a las señales continuas reales, pero no es posible realizar una Transformada de Fourier en tiempo continuo en un procesador, puesto que éste trabaja con valores finitos y cuantificados; por lo tanto, la señal se muestrea y de esa manera se realiza una Transformada de Fourier en tiempo discreto o DFT, definida en la ecuación 3.10, donde k es la frecuencia discreta.

$$x[n] = \sum_{k=<N>} X[k]e^{jk\Omega_0 n} \tag{3.10}$$

La transformada discreta de Fourier es simplemente una modificación de la serie de Fourier tradicional, sustituye las integrales por sumatorias de las muestras y el periodo en lugar de ser T (número real) será n, siendo n un número entero.

Sin embargo, esto es costoso de calcular, ya que el número de operaciones crece de manera cuadrática con el número de muestras, es decir, $No.Operaciones = No.Muestras^2$.

Mediante el algoritmo de Transformación Rápida de Fourier (FFT), este cálculo se reduce a una función de crecimiento logarítmica-lineal con el número de muestras, donde $No.Operaciones = No.Muestras \cdot log_2(No.Muestras)$. Es una técnica para realizar de una manera más eficiente y con un menor coste computacional, el cálculo de la Transformada Discreta de Fourier.

Para realizar el espectro de Fourier de los acelerogramas, se calcula la transformada rápida de Fourier para cada componente de la señal sísmica, la cual puede ser representada en escalas aritméticas, logarítmicas o una combinación de ellas.

Rutina Espectro de Frecuencia. Se accede a esta rutina en la opción *Espectro de Frecuencia* del menú *Procesamiento*. Permite hallar la Tranformada Discreta de Fourier de un vector o una matriz, calculada con el algoritmo de la Transformada Rápida de Fourier mediante la función *fft* de Matlab.

En primer lugar, se obtiene el intervalo de tiempo entre cada muestra a partir de los datos previamente recopilados de la frecuencia de muestreo y se crea el vector de los valores de frecuencia para realizar las gráficas.

```
dt =1./fm;
df =1./dur;
y = length(ch1);
freq =(0:y-1)'*df;
np2 = y/2+1;                %mitad de los datos mas uno
handles.ndat = np2;
per = freq(2:np2);
```

Se calculan los espectros de frecuencia para cada una de las componentes, utilizando la transformada rápida de Fourier de los datos mediante el comando *fft*. Se calcula el valor absoluto de la transformada para graficar los valores positivos.

```
Componente ns
xfns = fft(ch1);
xfnsam=abs(xfns)*dt;
nsff=xfnsam(2:np2);

Componente ew
xfew = fft(ch2);
xfewam=abs(xfew)*dt;
ewff=xfewam(2:np2);

Componente z
xfz = fft(ch3);
xfzam=abs(xfz)*dt;
zff=xfzam(2:np2);
```

Con los datos obtenidos en el dominio de la frecuencia, se trazan las gráficas de cada componente.

```
subplot(3,1,2), plot(per,nsff,'r-');
xlabel('Frecuencia (Hz)');
ylabel('Mod-FFT');
Title('ns');

subplot(3,1,3), plot(per,ewff,'r-');
xlabel('Frecuencia (Hz)');
ylabel('Mod-FFT');
Title('ew');

subplot(3,1,1), plot(per,zff,'r-');
xlabel('Frecuencia (Hz)');
ylabel('Mod-FFT');
Title('z');
```

En la figura 3.24 se pueden observar los espectros de Fourier calculados a partir de los datos de la figura 3.17.

3.3.7. Método de Nakamura. Según Alfaro et al.[9], a partir de microsismos registrados en superficie, es posible determinar la función de transferencia aproximada de las capas superficiales del suelo, mediante el método propuesto por Nakamura en 1989, el cual formula la función de transferencia del suelo como la relación entre el espectro de Fourier de la componente horizontal del microsismo y el espectro de su componente vertical; para llegar a esto, se basó en las siguientes suposiciones:

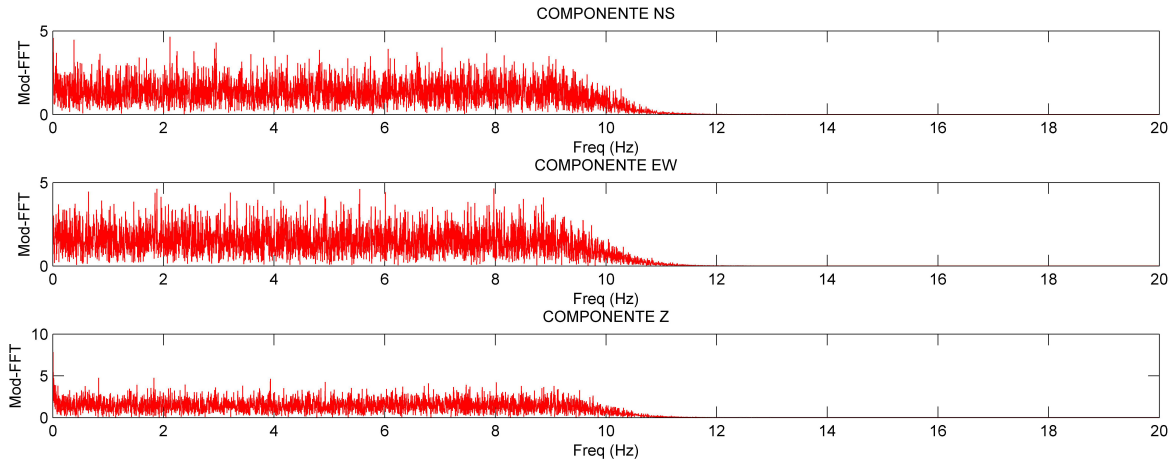
1. Las microtrepidaciones se propagan principalmente como trenes de onda Rayleigh.

2. El efecto de las ondas Rayleigh es igual para las componentes horizontales y verticales en superficie.

3. El ruido artificial se propaga principalmente como ondas Rayleigh.

[9]ALFARO, A. et al. Determinación de características dinámicas del suelo a partir de microtemblores [en línea]. Disponible en internet: `http://andresalfaro.com/doc_estudios/1999%20_alfaro_egozcue_ugalde.pdf`. p. 1-3.

Figura 3.24. Espectros de fourier.



4. Las componentes horizontal y vertical de las microtrepidaciones (en su origen) se consideran similares.

5. Las microtrepidaciones son amplificadas por las capas superficiales blandas del suelo acumuladas sobre un subestrato duro.

6. La componente vertical de las microtrepidaciones no es amplificada por las capas horizontales del suelo.

7. Se puede considerar que la componente horizontal de las microtrepidaciones es amplificada por la multireflexión de la onda P. Las ondas P son ondas longitudinales, esto significa que el suelo es alternadamente comprimido y dilatado en la dirección de propagación de la onda.

Debido a las grandes ventajas de este método en términos de logística y procesamiento, éste se ha difundido y es muy utilizado para obtener mayor información del subsuelo.

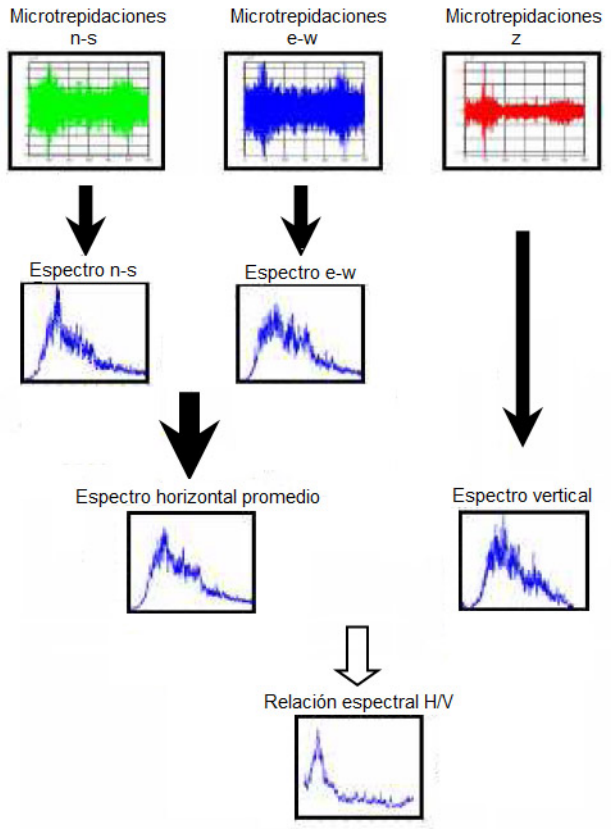
Rutina Método de Nakamura. En el programa de *Procesamiento de Datos* es posible obtener la relación entre los espectros de Fourier de las componentes horizontales y la componente vertical, a partir del acelerograma registrado.

Cuando se realiza el cálculo de los espectros de Fourier, aparece un cuadro de diálogo como el de la figura 3.25, en el que el usuario puede escoger si desea obtener las relaciones espectrales H/V.

Figura 3.25. Cuadro de diálogo para graficar cocientes espectrales

Para el cálculo de la relación espectral H/V según el método de Nakamura, es necesario encontrar el promedio de los espectros de Fourier horizontales, es decir, de las componentes *ns* y *ew*, luego se divide el espectro horizontal promedio entre el espectro de Fourier de la componente vertical z; este proceso se resume en la figura 3.26.

Figura 3.26. Procedimiento para obtener la relación espectral H/V.



Además, se encuentran las relaciones espectrales entre cada una de las componentes horizontales y la componente vertical. La rutina en Matlab es de la siguiente manera:

```
%Cocientes Espectrales
hprom=(nsff+ewff)/2;    %promedio componentes horizontales
nak=hprom./zff;        %promedio horizontal /componente z
nsz=nsff./zff;         %componentens / componentez
ewz=ewff./zff;         %componenteew / componentez
```

Posteriormente se grafican los resultados usando notación logarítmica con los comandos *plot* y *loglog*.

```
%proce relaciones espectrales
cocesp=figure(2);
set(cocesp,'Name','Cocientes Espectrales');

subplot(3,2,1), plot(per,nak,'k-');grid on;
title('Relación espectral H/V');
xlabel('Frec. (Hz)');
ylabel('H/V');
grid on;

subplot(3,2,4), loglog(per,nsz,'k-');grid on;
title('Representación logarítmica ');
xlabel('Frec. (Hz)');
ylabel('NS/V');
grid on;

subplot(3,2,5), plot(per,ewz,'k-') ;grid on;
title('Relación espectral EW/Z');
xlabel('Frec. (Hz)');
ylabel('EW/V');
subplot(3,2,6), loglog(per,ewz,'k-');grid on;
title('Representación logarítmica ');
xlabel('Frec. (Hz)');
ylabel('EW/V');
```

En la figura 3.27 se observan las gráficas de las relaciones espectrales de los datos de la figura 3.17.

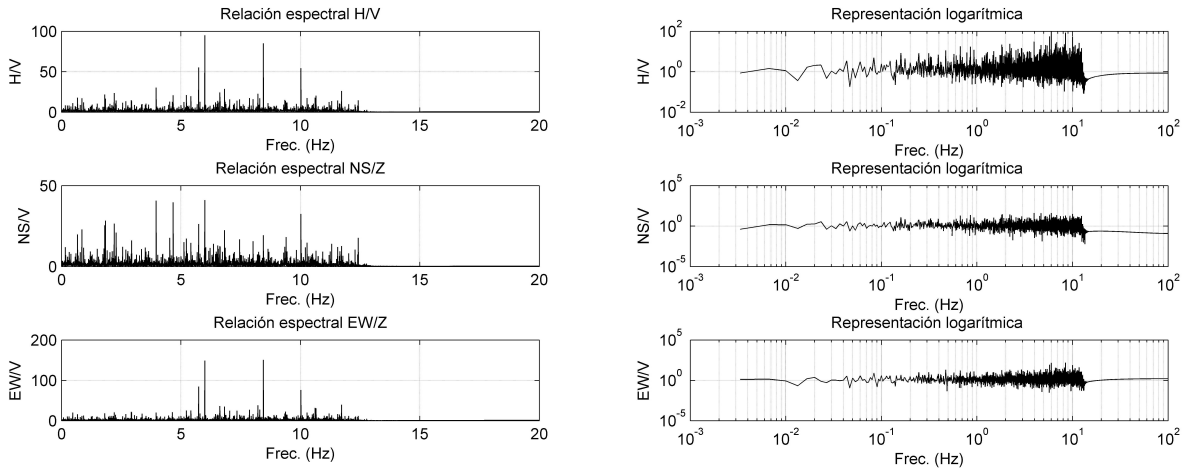
El cálculo de estas relaciones permite obtener un análisis de los períodos predominantes del suelo.

3.4. ESPECTROS DE RESPUESTA

Actualmente, el concepto de espectro de respuesta es una herramienta de gran utilidad en el área del diseño sismorresistente. En forma general, se define espectro como un gráfico de la respuesta máxima que produce una acción dinámica determinada en una estructura, expresada en términos de desplazamiento, velocidad, aceleración, o cualquier otro parámetro de interés.

Si se somete una serie de osciladores a la acción de un mismo terremoto (utilizando un registro de aceleraciones o acelerograma), cada uno de ellos exhibirá una respuesta diferente, la cual puede representarse, por ejemplo, a través de la historia de desplazamientos. Una vez calculada la respuesta de los osciladores es posible determinar el

Figura 3.27. Gráfica de los cocientes espectrales.



máximo (en valor absoluto, dado que el signo no tiene importancia) de cada uno de ellos y representarlos en un gráfico en función del periodo de vibración, para obtener así un espectro de respuesta. La respuesta máxima de cada oscilador con periodo T representa un punto del espectro.

Existen varios tipos de espectros, los cuales presentan características diferentes y se utilizan con distintos objetivos, los espectros más comunes son:

- *Espectros de respuesta elástica:* representan parámetros de respuesta máxima para un sismo determinado. Se utilizan fundamentalmente para estudiar las características del sismo y su efecto sobre las estructuras.
- *Espectros de respuesta inelástica:* son similares a los anteriores, pero en este caso se supone que la estructura puede experimentar deformaciones por acción del terremoto, es decir, tiene comportamiento no-lineal.
- *Espectros de diseño:* las construcciones no pueden diseñarse para resistir un terremoto en particular en una zona dada, puesto que el próximo terremoto probablemente presentará características diferentes. Por lo tanto, los espectros de respuesta elástica o inelástica, descritos anteriormente, no pueden utilizarse para el diseño sismorresistente; por esta razón, el diseño o verificación de las construcciones sismorresistentes se realiza a partir de espectros que consideran el efecto de varios terremotos.

Se han desarrollado varias metodologías para obtener los espectros de diseño, basadas en

procedimientos estadísticos. El procedimiento más usual es considerar el valor promedio más la desviación estándar de los espectros de respuesta de varios terremotos representativos, de modo que se tiene en cuenta la mayor o menor dispersión de los datos y conduce a resultados confiables.

3.4.1. Microtrepidaciones y espectros de respuesta. Con el fin de dar cumplimiento a lo expresado en las Normas Colombianas de Diseño y Construcción Sismorresistente NSR-98 (AIS, 1998) que expresa: "las capitales de departamentos y las ciudades de más de 100.000 habitantes, localizadas en las zonas de amenaza sísmica intermedia y alta, con el fin de tener en cuenta el efecto que sobre las construcciones tenga la propagación de la onda sísmica a través de los estratos de suelo subyacentes, podrán armonizar las reglamentaciones municipales de ordenamiento del uso de la tierra, con un estudio o estudios de microzonificación sísmica"[10].

Dentro de las actividades programadas para el desarrollo de la microzonificación de ciudades se contemplan las campañas de microtrepidaciones, con las cuales se obtienen registros para la elaboración de un mapa de isoperíodos. Es con la información que provee este mapa, así como con registros de sismos seleccionados y mediante la modelación de las zonas geotécnicas que se determinan los espectros de respuesta de diseño, objetivo fundamental de una microzonificación.

El uso de registros de microtrepidaciones para la evaluación de la respuesta dinámica de suelos es muy atractivo, debido a su bajo costo y a la facilidad y rapidez para obtener mucha información. Sin embargo, la información proveniente de registros de microtrepidaciones corresponde al rango de deformaciones bajas del suelo, el cual sirve para estimar diferencias relativas entre suelos, pero no para determinar la respuesta nolineal a altas deformaciones del suelo, como suele suceder durante sismos intensos.

La determinación de períodos a grandes deformaciones, sigue siendo dependiente de la disponibilidad de registros de movimiento fuerte o de una caracterización detallada del comportamiento dinámico de los suelos y de una posterior modelación. De todas formas, los resultados de las microtrepidaciones ofrecen la posibilidad de calibrar los modelos de comportamiento dinámico de suelo a bajas deformaciones, para luego ser usados en modelos de grandes deformaciones. Adicionalmente, las microtrepidaciones han demostrado ser útiles para determinar zonas de distinto comportamiento dinámico. En general, se ha encontrado que las microtrepidaciones son una herramienta útil para adelantar estudios de dinámica de suelos y microzonificación sísmica.

Teniendo en cuenta que la Universidad de Nariño - Centro de Investigación en Ingeniería Sismológica, cuenta con equipos como el acelerógrafo CUSP-3C y el obtenido a partir de esta investigación, además de estudios de microzonificación sísmica preliminares

[10]Normas Colombianas de Diseño y Construcción Sismorresistente NSR-98. Asociación Colombiana de Ingeniería Sísmica (AIS), 1998

realizadas en la ciudad de San Juan de Pasto por INGEOMINAS y la Universidad de Nariño, e innumerables investigaciones de registro de microtrepidaciones tendientes a obtener la misma, se dispone de las herramientas necesarias para continuar con este tipo de investigaciones, aspecto que se tiene en cuenta en las recomendaciones de este proyecto de investigación.

Cabe resaltar que el equipo EAR-1 no esta diseñado para registrar las señales en los rangos de frecuencia de sismos fuertes, su uso está limitado para realizar registros de microsismos.

4 CONSTRUCCIÓN EXTERNA DEL EQUIPO DE ADQUISICIÓN Y REGISTRO DE MICROSISMOS

4.1. DISEÑO

Para proteger el sistema de posibles impactos, fenómenos ambientales como lluvias e incluso, evitar la pérdida de sus componentes, se diseña una caja con ayuda del software Solid Edge V.17 versión académica provista por el Servicio Nacional de Aprendizaje (SENA), el cual permite realizar un diseño tridimensional que facilita el proceso de elaboración.

Teniendo en cuenta las medidas de los distintos componentes y una separación adecuada entre ellos, las medidas finales para la caja son: 30 cm de ancho por 30 cm de largo y una altura de 15 cm.

En las figuras 4.1, 4.2 y 4.3, se muestran las imágenes de la ventana del software utilizado y el diseño tanto de la base donde se colocan los componentes como de la respectiva tapa.

Figura 4.1. Diseño de la protección exterior (componentes)

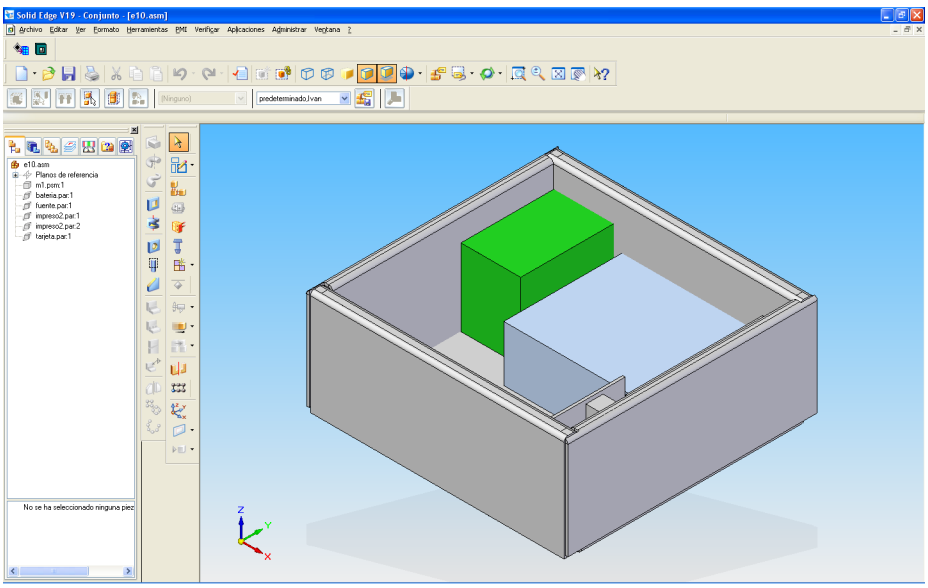


Figura 4.2. Diseño de la protección exterior (tapa)

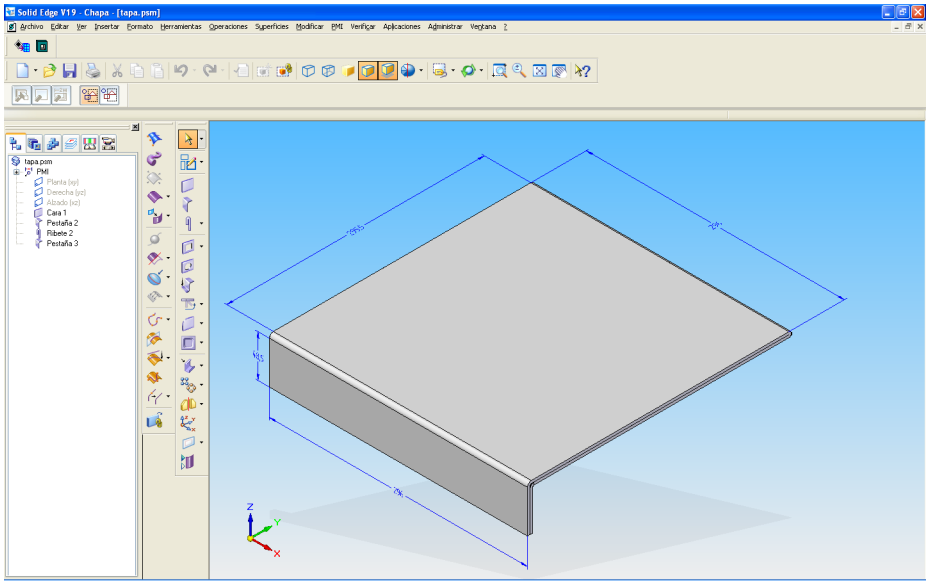
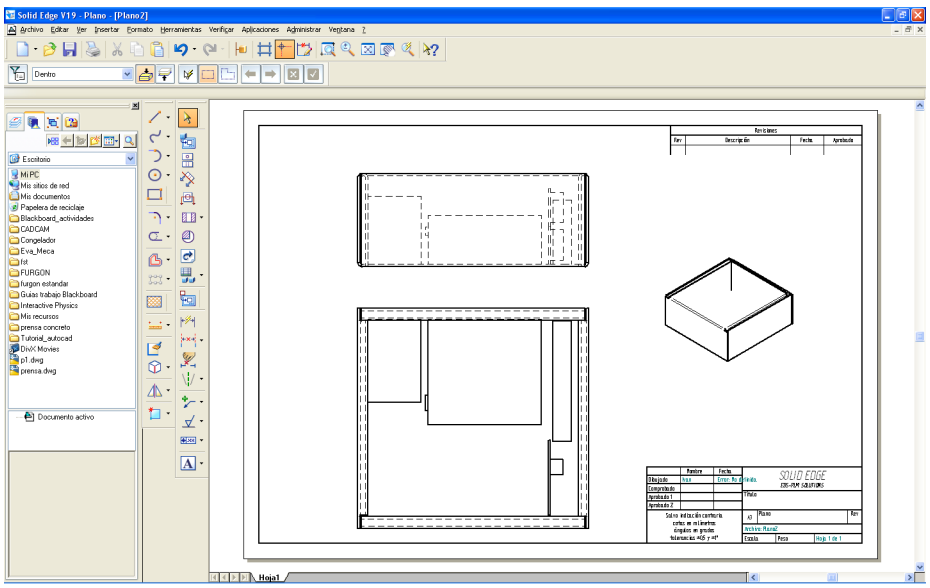
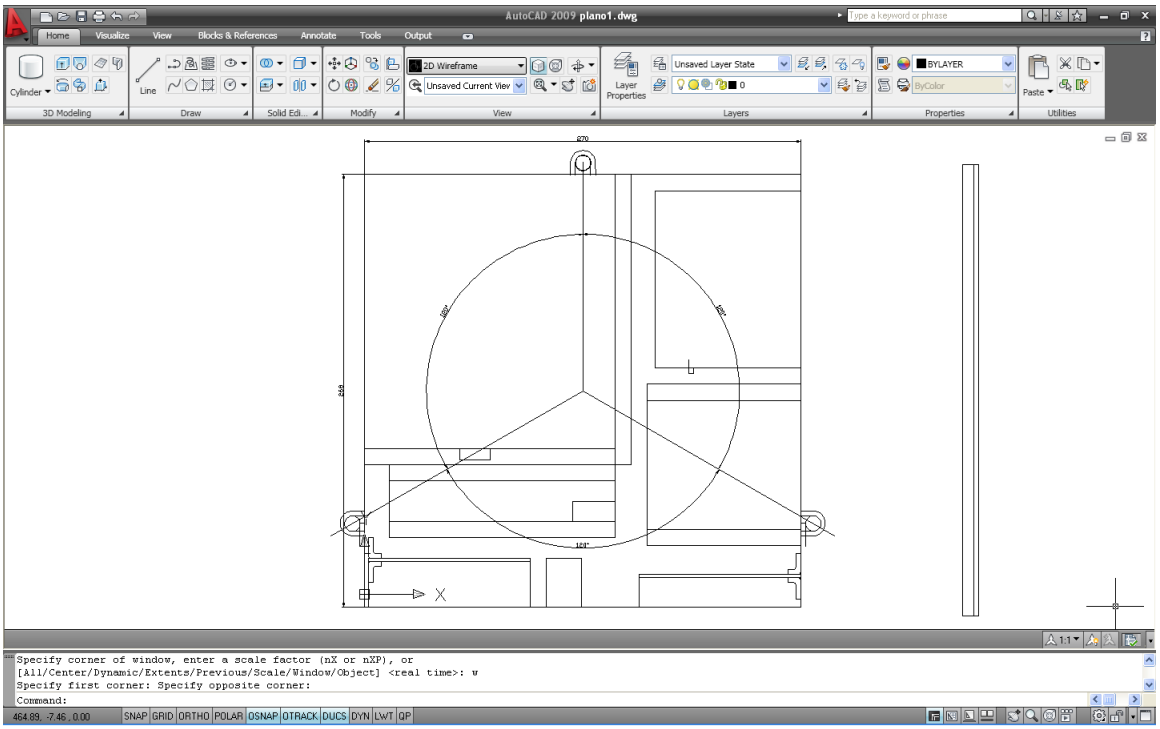


Figura 4.3. Vista lateral derecha, vista superior del equipo



Con ayuda del software Autocad V2005, igualmente provisto en versión académica por el Servicio Nacional de Aprendizaje (SENA), se calcula la posición de 3 niveladores que van adheridos a las paredes de la caja como se muestra en la figura 4.4. El mismo procedimiento se realiza para los niveladores que se ubican en la caja metálica donde se encuentra el acelerómetro.

Figura 4.4. Ubicación de los niveladores del equipo



4.2. SELECCIÓN DEL MATERIAL

Para escoger el material a utilizar en la protección exterior, se tiene en cuenta que debe ser un material duro, pero a la vez flexible, resistente a impactos, fácil de manipular, impermeable, no corrosivo y que no afecte los componentes del sistema.

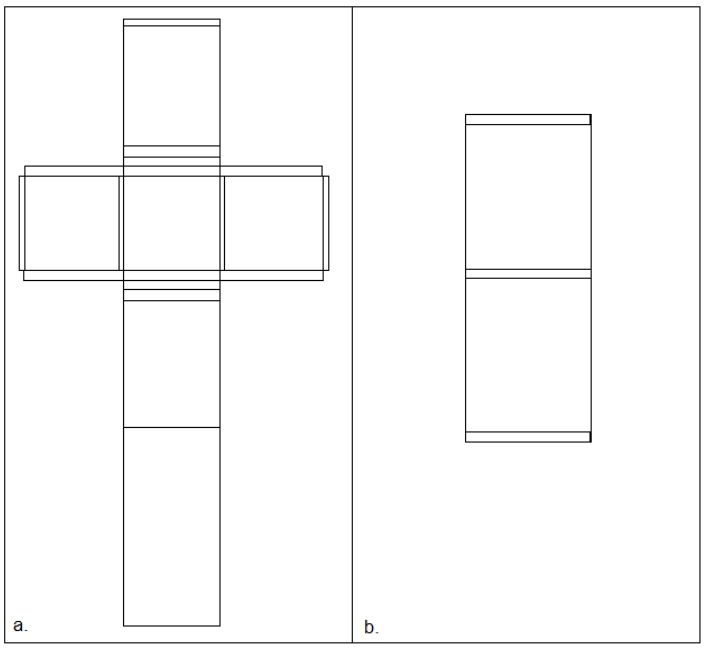
Teniendo en cuenta estas características, se descarta cualquier tipo de metal para evitar problemas de campos electromagnéticos no deseados y de acuerdo a la disponibilidad de material en la ciudad, se encuentra que la lámina de acrílico de 3 mm de espesor cumple con los requisitos anteriormente especificados.

4.3. PROCEDIMIENTO

A continuación se describe el procedimiento para llevar a cabo el desarrollo y fabricación de la maleta.

Trazado y corte. Una vez obtenida la lámina se traza sobre ella el desarrollo plano del diseño, es decir, el dibujo bidimensional del diseño volumétrico hecho en Solid Edge, como se observa en la figura 4.5, se realiza el corte en los bordes con ayuda de una segueta.

Figura 4.5. Desarrollo del plano de la maleta y de la tapa



Doblado. Para doblar el material se realiza un procedimiento previo llamado grafado. El grafado consiste en repujar las líneas a doblar por el lado contrario al doblez, con el fin de facilitar la deformación del material en el proceso de doblado. Para este procedimiento se utiliza un grafador de mano.

Posteriormente se procede a doblar el material. Se utiliza una dobladora de acrílico de bajo espesor, la cual consta de una resistencia con un recubrimiento de fibra que proporciona el calor necesario para que el material se doble fácilmente sin dañarse.

Una vez doblado se pegan los lados con ayuda de Super Bonder, Metil Metacrilato líquido y se corrigen las fallas con ayuda de resina de acrílico autopolimerizable.

Perforación. Se realiza una medición previa de todos los elementos que van incrustados

en la maleta, como: leds, conectores y ventilación. Luego se traza en el acrílico y se perfora con ayuda de un taladro y un motor tool.

Relleno. Debido a que la maleta tiene doble fondo se rellena con espuma de poliuretano, la cual es estable dimensionalmente, no se expande, ni se contrae o deforma en el tiempo, se adhiere a todo tipo de materiales, bajo peso, excelente capacidad aislante. Esto hace que la caja tenga mayor resistencia y sea más estable.

Debido a las complicaciones que presenta la mezcla y aplicación de este relleno, se recomienda que sea hecho por personas expertas en el tema.

Niveladores. Para realizar una correcta toma de datos, es necesario que el acelerómetro Crossbow se encuentre nivelado sobre la superficie que se coloca en el momento de registrar los microsismos.

Se elaboran 3 niveladores utilizando tornillos de rosca fina para llave hexagonal y bujes. Se aseguran en las paredes de la caja metálica que recubre al acelerómetro y mediante un indicador de nivel de tipo "ojo de pollo" colocado en la parte superior de la caja, se observa si la superficie del acelerómetro se encuentra nivelada o si necesita mayor altura de un lado u otro, para realizar una medición más precisa.

Igualmente se disponen de niveladores en la parte exterior de la maleta, para cuando se desee realizar pruebas con el acelerómetro en el interior de la misma.

Separadores. Es importante que los componentes se encuentren bien sujetos para evitar daños, interferencias o posibles variaciones en el registro por movimientos internos.

Los separadores son trozos de acrílico que van pegados a la base y paredes de la maleta. Su función es sujetar los componentes, todos los separadores miden 1cm de alto por 1cm de ancho y su longitud se define dependiendo de la parte que van a sujetar.

Los separadores se adhieren usando superbonder y metilmetacrilato.

Tapa. La tapa se une con el resto de la maleta por medio de una varilla de aluminio, que pasa a través de los orificios ubicados en los extremos de la misma. Se utilizan chapas en la parte frontal para asegurarla con la base de la maleta.

Se coloca una espuma de 5 cm de espesor en la parte interna de la tapa para que ayude a los separadores a mantener fijos los elementos y para amortiguar el cierre de la tapa.

Pintura. Después de los procedimientos anteriores, se debe pulir y limar los bordes de la caja y tapa, incluyendo las perforaciones y posibles restos de espuma de poliuretano, para lograr una superficie totalmente lisa, sin residuos y así aplicar la pintura.

Primero se aplica una base para plásticos y se deja secar por 20 minutos aproximadamente, luego se aplican 3 capas de pintura en aerosol dejando un lapso de tiempo de aproximadamente 15 minutos entre cada una.

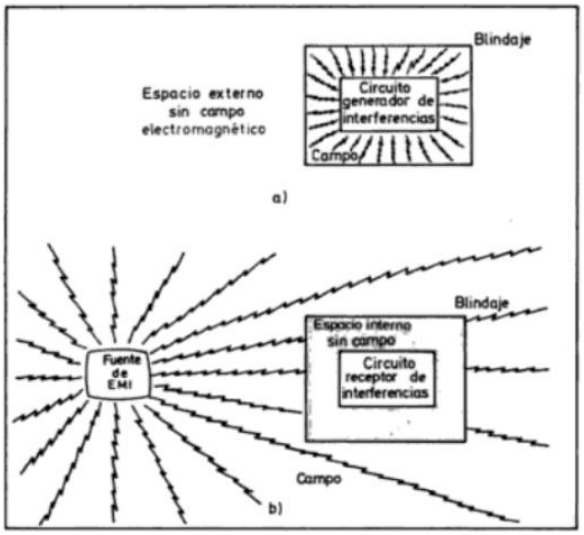
El color que se aplica es amarillo, como refiere la norma de instrumentación sísmica.

Ubicación de los elementos. Se procede a colocar cada uno de los elementos del sistema dentro de la maleta, de manera que queden fijos y que ninguna conexión quede forzada.

4.4. BLINDAJE ELÉCTROMAGNÉTICO

Como lo afirman Balcells y Daura[1], un blindaje es una superficie metálica dispuesta entre dos regiones del espacio que se utiliza para atenuar la propagación de los campos eléctricos, magnéticos y electromagnéticos. Un blindaje sirve tanto para no dejar salir el flujo de los campos de la zona encerrada por él, como para evitar que en una zona protegida por el mismo entre campo alguno; esto se ejemplifica en la figura 4.6.

Figura 4.6. Efecto del blindaje



Fuente: BALCELLS, Josep; DAURA, Francesc y otros. Interferencias electromagnéticas en sistemas electrónicos. Barcelona : Marcombo, 1991. p. 85.

La captación de un campo eléctrico o magnético es tratada como un acoplamiento capacitivo o inductivo respectivamente. El modo de bloquear el acoplamiento capacitivo consiste en encerrar el circuito o el conductor que se quiere proteger dentro de un blindaje metálico. Este es el llamado blindaje electrostático o de Faraday (jaula de Faraday), el cual también contribuye a reducir el acoplamiento inductivo.

[1]BALCELLS, Josep; DAURA, Francesc y otros. Interferencias electromagnéticas en sistemas electrónicos. Barcelona : Marcombo, 1991. p. 85.

En el equipo de registro de microtrepidaciones, se coloca una caja metálica para cubrir el acelerómetro Crossbow, proporcionando el efecto jaula de Faraday sobre él.

La jaula de Faraday es un recubrimiento metálico con la característica de aislar el espacio interior de campos electromagnéticos externos, de tal modo que las descargas que se producen en el exterior de la jaula no afectan el interior y simultáneamente aisla el exterior de los campos y ondas electromagnéticas que se generan en el interior. Se evita así que se puedan producir chispas (descargas eléctricas) en el exterior, en zonas no controladas donde den lugar a incendios o averias de aparatos electrónicos[2].

Según Heckert[3], cuando el conductor está sujeto a un campo electromagnético externo, se polariza, de manera que queda cargado positivamente en la dirección en que va el campo electromagnético, y cargado negativamente en el sentido contrario. Puesto que el conductor se ha polarizado, este genera un campo eléctrico igual en magnitud pero opuesto en sentido al campo electromagnético, por lo tanto, la suma de ambos campos dentro del conductor será aproximadamente igual a cero.

Cada material posee una profundidad característica de atenuación de la onda electromagnética que intenta desplazarse por ese medio, que determina la distancia que recorre la onda hasta que su campo eléctrico se reduce en un factor $1/e = 0{,}367$ (siendo $e = 2{,}7182$), es decir, cuando el campo eléctrico se reduce en un 36,7 %. La densidad de potencia, o cantidad de energía que atraviesa un área determinada por segundo de una onda electromagnética, viene dada por el cuadrado del campo eléctrico, así que en esa misma distancia, la potencia de la onda disminuye en un factor $1/e^2 = 0{,}135$, un 13,5 %.

Un 13,5 % puede ser una cantidad pequeña, pero no es nula. La jaula de Faraday juega con el espesor de las paredes produciendo una atenuación suficiente para lograr que la potencia que consigue atravesarla sea despeciable. Cuanto mayor sea el espesor, menos energía traspasará la jaula e interferirá con el circuito.

La profundidad característica depende además de la frecuencia de la onda incidente, de forma que una jaula puede ser efectiva para unas frecuencias, pero no para otras. En general, cuanto mayor es la frecuencia, menor es la profundidad característica (se atenúa más rápido).

Para el aluminio, una frecuencia de 60Hz necesita una pared con un espesor de alrededor de 1 cm para ser atenuada un 13 %; sin embargo, para 3Ghz, se consigue la misma atenuación con menos de 2 micras (0.002 milímetros). Se considera que el hierro es el mejor de todos los materiales conductores para hacer una jaula de Faraday.

Usos. El primero en emplear este dispositivo para verificar que el campo eléctrico en el

[2]Antonio Gros. Ceuta (España)

[3]HECKERT A., Paul. Understanding the faraday cage. Disponible en internet: `http://electricitymagnetism.suite101.com/article.cfm/understanding_the_faraday_cage`

interior de un conductor hueco es nulo, fue Faraday en 1936. Para ver qué sucedía con los conductores huecos y los campos electrostáticos, construyó una habitación cubierta con papel de aluminio y aplicó descargas de alto voltaje con un generador electrostático sobre el exterior de la sala. Dentro de la habitación colocó un electroscopio, de manera tal que quedaba demostrado que no existía un campo eléctrico en el interior de la misma.

Actualmente, este tipo de blindaje de campo electromagnético tiene diversas aplicaciones, se puede mencionar por ejemplo, la protección de equipos electrónicos delicados, tanto en laboratorios como en ciertas construcciones.

5 RESPUESTA INSTRUMENTAL: PRUEBAS Y CALIBRACIÓN

5.1. PRUEBA No. 1. PRUEBA DE FUNCIONAMIENTO DEL SISTEMA DE ADQUISICIÓN DE DATOS

.

Dentro del sistema de adquisición se involucra todo el desarrollo, desde el instante en el que la señal analógica procedente del amplificador de instrumentación es digitalizada en intervalos de 5 milisegundos (200mps), hasta su almacenamiento.

Mediante un generador de ondas se introduce una señal senoidal de amplitud aproximadamente igual a un voltio, a la entrada de los amplificadores de instrumentación dispuestos para recibir las señales de salida provenientes del acelerómetro, correspondientes a los ejes x (componente norte-sur), y (componente este-oeste) y z (componente vertical).

Despúes de la etapa de amplificación la señal pasa através del filtro anti-aliasing y es adquirida por un lapso de tiempo de 10 segundos mediante la tarjeta de adquisición de datos utilizando los canales 1, 2 y 3, a una tasa de 200 muestras por segundo, esto se realiza con el programa desarrollado de adquisición y registro de señales microsísmicas.

Se realiza un barrido de frecuencias de la onda senoidal desde 0.4 a 100Hz. Para cada valor de frecuencia se almacenan los datos de la señal de entrada y los obtenidos a las salidas de la etapa de acondicionamiento en un archivo de texto.

Los valores de amplitud y de frecuencia de las señales emitidas por el generador, se verifican con un osciloscopio marca Protek como el de la figura 5.1, perteneciente al laboratorio de electrónica de la Universidad de Nariño; estas señales también se adquieren utilizando el canal 4 de la tarjeta NI USB-9215, de esta forma se comprueba su similitud.

Se grafican los datos en el tiempo para obtener el valor de la amplitud pico de la señal de entrada y de cada una de las salidas, como se ejemplifica en la figura 5.2 para una frecuencia de 1Hz.

Esto se realiza para señales de diferentes frecuencias y se obtienen los datos que se indican en el cuadro 5.1. En las columnas 3, 4 y 5 están los valores de salida en voltaje correspondientes a los canales 1, 2 y 3 respectivamente, del módulo de entradas analógicas de la tarjeta de adquisición de datos y en las columnas 6, 7 y 8, los valores reales de ganancia, los cuales se aproximan a tres, el valor teórico asignado durante la etapa de diseño.

Figura 5.1. Osciloscopio de prueba del laboratorio de electrónica de la Universidad de Nariño

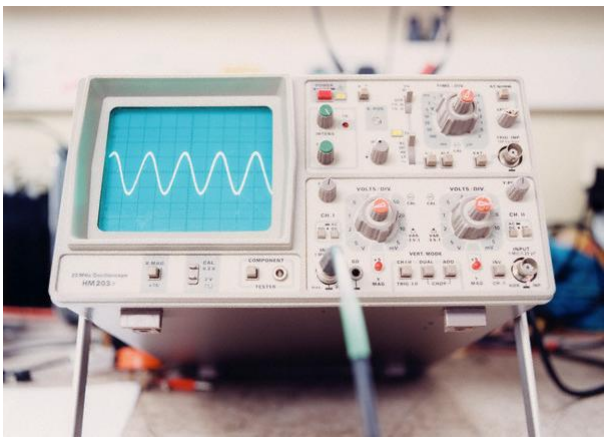
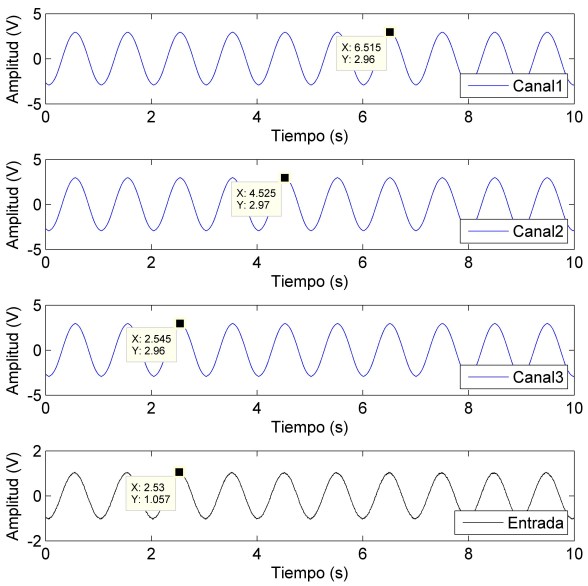


Figura 5.2. Gráficas de señales de salida obtenidas en el sistema de adquisición de datos usando como entrada una señal senoidal de amplitud pico igual a 1 voltio



El valor de la ganancia para cada uno de los ejes en las diferentes frecuencias, se calcula mediante la relación entre el valor de salida y el de entrada. En la figura 5.3 se muestran las gráficas de la ganancia con respecto a la frecuencia para cada uno de los ejes.

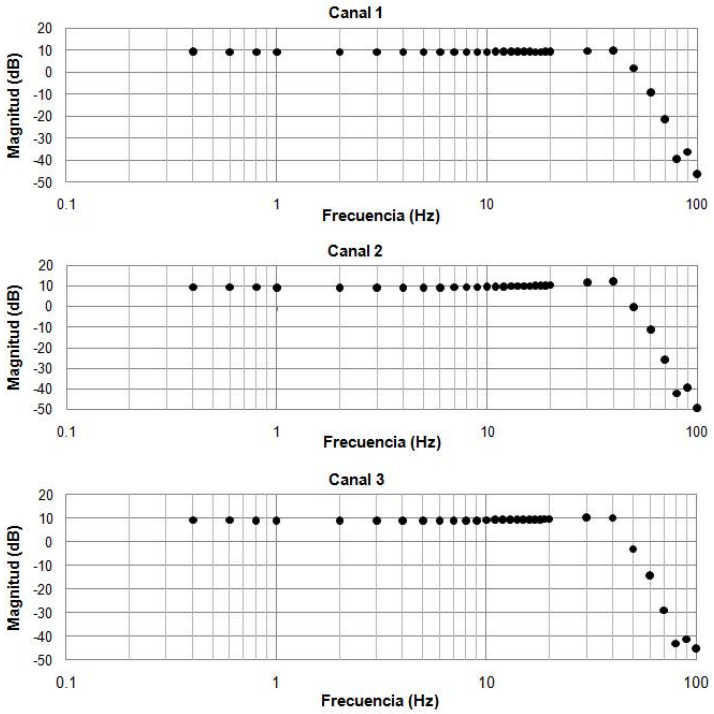
Cuadro 5.1. Datos obtenidos mediante prueba del sistema de adquisición con generador de señales

Frec.(Hz)	Entrada(V)	x(V)	y(V)	z(V)	Ganancia x	Ganancia Y	Ganancia Z
0.4	1.0309	2.9391	2.9549	2.9437	2.8510	2.8663	2.8555
0.6	1.0362	2.9440	2.9604	2.9482	2.8412	2.8570	2.8452
0.8	1.0364	2.9437	2.9608	2.9474	2.8403	2.8568	2.8439
1	1.0569	2.9613	2.9798	2.9660	2.8019	2.8194	2.8063
2	1.0330	2.9164	2.9371	2.9202	2.8232	2.8433	2.8269
3	1.0317	2.9035	2.9283	2.9064	2.8143	2.8383	2.8171
4	1.0375	2.9153	2.9449	2.9186	2.8099	2.8385	2.8131
5	1.0396	2.9066	2.9419	2.9085	2.7959	2.8298	2.7977
6	1.0383	2.9024	2.9458	2.9064	2.7953	2.8371	2.7992
7	1.0364	2.9055	2.9622	2.9155	2.8035	2.8582	2.8131
8	1.0304	2.9099	2.9783	2.9212	2.8240	2.8904	2.8350
9	1.0325	2.8963	2.9904	2.9220	2.8051	2.8963	2.8300
10	1.0288	2.9188	3.0385	2.9605	2.8371	2.9534	2.8776
11	1.0322	2.9364	3.0793	2.9926	2.8448	2.9832	2.8992
12	1.0334	2.9645	3.1227	3.0232	2.8687	3.0218	2.9255
13	1.0307	2.9817	3.1452	3.0326	2.8929	3.0515	2.9423
14	1.0301	2.9829	3.1474	3.0144	2.8957	3.0554	2.9263
15	1.0294	2.9607	3.1441	3.0039	2.8761	3.0543	2.9181
16	1.0306	2.9325	3.1548	3.0071	2.8454	3.0611	2.9178
17	1.0310	2.9287	3.1859	3.0375	2.8406	3.0901	2.9462
18	1.0347	2.9356	3.2285	3.0733	2.8372	3.1202	2.9702
19	1.0308	2.9480	3.2704	3.1076	2.8599	3.1727	3.0147
20	1.0252	2.9633	3.3111	3.1269	2.8905	3.2297	3.0500
30	1.0302	3.0440	3.8063	3.4033	2.9548	3.6947	3.3035
40	0.8719	2.6990	3.5116	2.7622	3.0955	4.0275	3.1680
50	1.0310	1.2520	0.9930	0.7124	1.2144	0.9631	0.6910
60	1.0314	0.3511	0.2746	0.1985	0.3404	0.2662	0.1925
70	1.0319	0.0851	0.0520	0.0361	0.0825	0.0504	0.0350
80	1.0633	0.0111	0.0082	0.0072	0.0104	0.0077	0.0068
90	1.0466	0.0158	0.0112	0.0088	0.0151	0.0107	0.0084
100	1.0479	0.0049	0.0036	0.0056	0.0047	0.0034	0.0053

En las gráficas se aprecia el valor de la ganancia en dB con respecto a la frecuencia; es posible observar que en el rango de frecuencias entre DC y 20Hz la ganancia permanece constante y es igual a 10dB, en 50Hz sufre una atenuación igual a 3dB, esta es la frecuencia de corte que se tuvo en cuenta en el diseño del filtro anti-aliasing. Para frecuencias mayores a 50Hz la señal se atenúa evitando muestrear datos a frecuencias iguales o mayores a la frecuencia de muestreo determinada (200Hz).

Mediante el comportamiento observado en frecuencia se caracteriza la respuesta del sistema de adquisición, lo cual demuestra conformidad con los parámetros de diseño seleccionados.

Figura 5.3. Gráficas de ganancia versus frecuencia para el sistema de adquisición de datos



5.2. PRUEBA No. 2. RESPUESTA EQUIPO DE ADQUISICIÓN Y REGISTRO DE MICROSISMOS ANTE SEÑALES CONOCIDAS

Para determinar la respuesta en frecuencia del acelerómetro Crossbow CXL04GP3-R en conjunto con el sistema de adquisición desarrollado, se realizan pruebas generando vibraciones a diferentes frecuencias y se analizan los datos en el dominio del tiempo y de la frecuencia.

Las señales de salida del acelerómetro se conectan a la entrada de la etapa de acondicionamiento, y se asigna para el eje x el canal de entrada 1 de la tarjeta de adquisición de datos, para el eje y el canal 2 y para el eje z el canal 3.

Se utiliza un montaje experimental como el de la figura 5.4, en el que se emplea un generador de ondas para inyectar una señal senoidal a un parlante de 8Ω y 160W, sobre el que se ubica el sensor, teniendo en cuenta que su eje x apunte hacia el norte cardinal y que se encuentre nivelado.

Se toman datos para frecuencias de señales dentro del rango de interés ($DC - 20Hz$), con los cuales se generan archivos de texto que almacenan los datos de las señales de

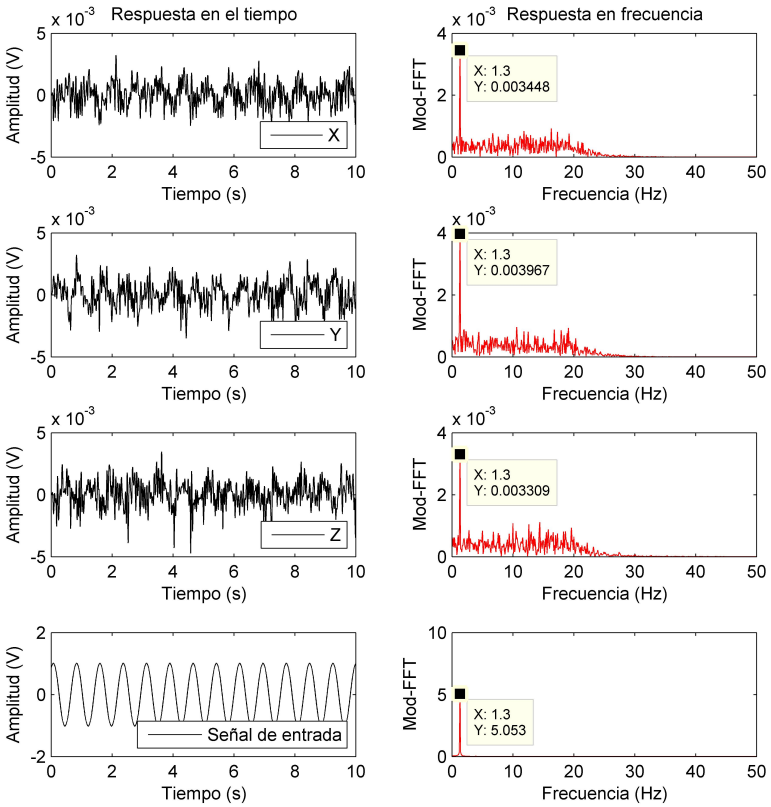
Figura 5.4. Montaje experimental No. 1: Respuesta equipo de adquisición y registro de microsismos ante señales conocidas.

salida del sensor y de la señal del generador, en unidades de voltaje. La adquisición se realiza durante 10 segundos para cada uno de los eventos y a 200 muestras por segundo.

Se realiza corrección de línea de base y se gráfica la amplitud en función del tiempo usando los datos de cada señal. Mediante el cálculo de la transformada discreta de Fourier (DFT), utilizando el algoritmo de la transformada rápida de Fourier (FFT), se encuentra la representación en el plano de la frecuencia. Como ejemplo, en las figuras 5.5 y 5.6 se muestran las respuestas para las frecuencias de 1 y 12Hz respectivamente.

En las gráficas, se puede observar que la señal captada por el equipo en cada uno de los ejes, corresponde a la frecuencia a la cual se fija la señal senoidal utilizada para generar la vibración, esto sucede igualmente para cada una de las frecuencias del rango de interés, lo cual implica una adecuada respuesta del instrumento de medición en cuanto a captar correctamente la frecuencia de vibración a la cual esté sometido.

Figura 5.5. Datos obtenidos para una señal de 1Hz.



5.3. CALIBRACIÓN

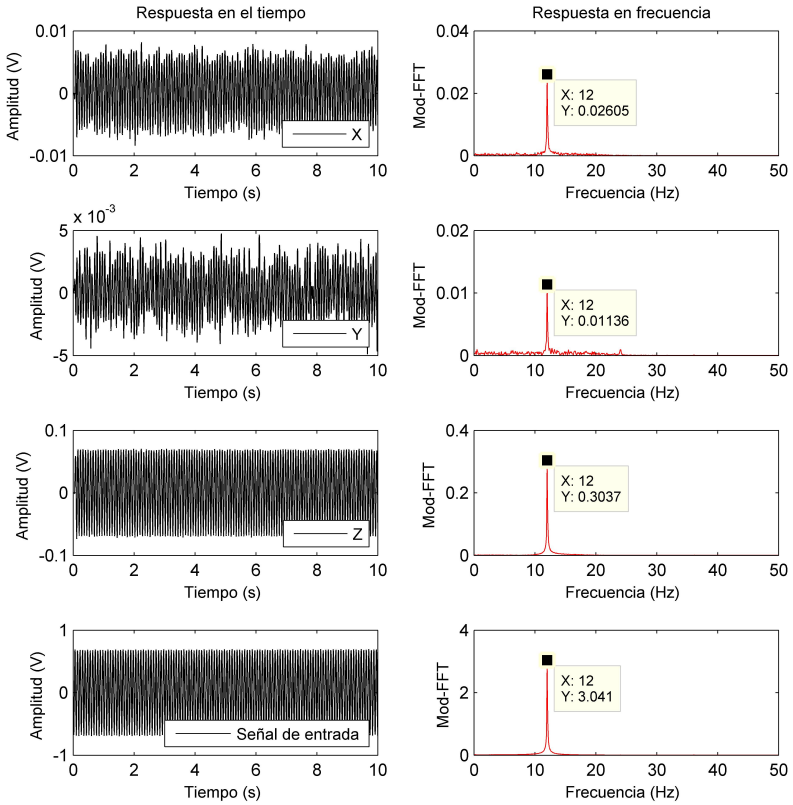
La calibración es el conjunto de operaciones con las que se establece, en condiciones específicas, la correspondencia entre los valores indicados por un instrumento, equipo o sistema de medida, y valores conocidos correspondientes a una magnitud de medida o patrón, procediendo a su ajuste para expresar su correspondencia.

Para calibrar un instrumento es necesario disponer de uno de mayor precisión, el cual proporciona el valor verdadero que se emplea para la comparación de los valores indicados por el instrumento sometido a calibración.

La calibración de un equipo se realiza con el fin de:

- Verificar el buen funcionamiento del equipo.

Figura 5.6. Datos obtenidos para una señal de 12Hz.



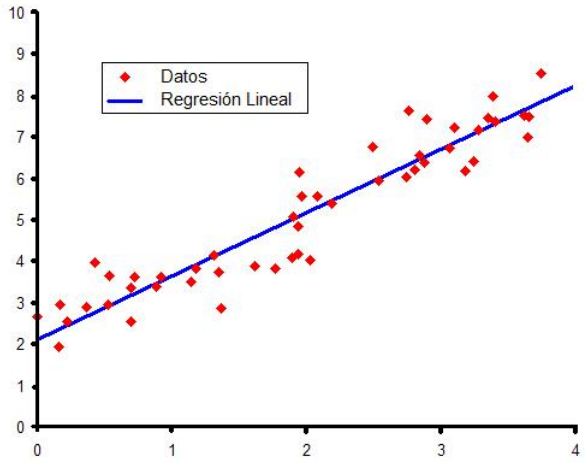
- Responder a los requisitos establecidos en las normas de calidad.
- Garantizar la fiabilidad y trazabilidad de las medidas.

5.3.1. Calibración lineal. Una de las opciones para realizar la calibración del equipo es la calibración lineal. El modelo de línea recta es un modelo muy usado debido a su soporte teórico y a su simplicidad. Según Riu y Boqué[1], consiste en encontrar la recta de calibrado que mejor se ajuste a una serie de n puntos experimentales, donde cada punto se encuentra definido por una variable x (variable independiente, equipo patrón) y una variable y (variable dependiente, equipo a calibrar). La recta de calibrado se encuentra definida por una ordenada en el origen (b_0) y una pendiente (b_1), a través de la ecuación 5.1, como se observa en la figura 5.7.

[1]RIU, Jordi; BOQUÉ, Ricard. Calibración Lineal [en línea]. Disponible en internet: `http://www.quimica.urv.es/quimio/general/callin.pdf`. p. 1-3.

$$y = b_0 + b_1 \cdot x \tag{5.1}$$

Figura 5.7. Regresión lineal. Método de mínimos cuadrados.



La calibración se reduce entonces, a encontrar las estimaciones de los coeficientes de la recta de calibrado (la ordenada en el origen y la pendiente), y a asegurar que la recta encontrada se ajusta correctamente a los puntos experimentales (es decir, asegurar que no hay falta de ajuste).

El método más universalmente empleado para encontrar los coeficientes de la recta de calibrado es el método de mínimos cuadrados, este método busca una recta de calibrado que haga que la suma de los cuadrados de las distancias verticales entre cada punto experimental y la recta de calibrado sea mínima .

Para obtener los datos que permitan realizar la calibración lineal, se somete el equipo desarrollado y el instrumento patrón, es decir el acelerómetro CUSP-3C, a vibraciones de un parlante que recibe una señal senoidal de amplitud y frecuencia conocidas.

Se realiza el montaje de la figura 5.8, donde se ubica el acelerógrafo CUSP - 3C y el acelerómetro Crossbow sobre una superficie plástica que a su vez se coloca en la parte superior del parlante.

Mediante el generador de señales se inyecta una señal senoidal al parlante, para que genere vibraciones a diferentes frecuencias dentro del rango de interés (DC-12Hz). Se registran los datos captados por los dos equipos y se procede a realizar una comparación entre ellos, de modo que sea posible obtener una curva de calibración para el equipo de adquisición de microsismos.

Antes de comparar es necesario realizar corrección de línea de base, tanto a los datos del

Figura 5.8. Montaje experimental No. 2

acelerógrafo CUSP - 3C como a los del EAR-1, nombre del equipo desarrollado. Además, para obtener una amplitud en unidades correspondientes a las del instrumento patrón, los datos se convierten de unidades de voltaje a unidades de aceleración (mg).

Se tabula y ordena los datos para cada uno de los ejes usando el programa Microsoft Office Excel y se realiza la regresión lineal de los datos para cada una de las frecuencias, teniendo en cuenta que x, la variable independiente corresponde a los datos del equipo patrón, acelerografo CUSP-3C y y, corresponde a los datos obtenidos con el sistema de adquisición y registro del equipo EAR-1.

Para describir el procedimiento se utilizan los datos correspondientes a 12Hz, como se muestra en la figura 5.9.

En las gráficas se observa un rizado no deseado en la señal del equipo EAR - 1 que no se presenta en el equipo patrón, por lo cual se aplica un filtro pasa bajas con frecuencia de corte en 20Hz para eliminar el rizado obteniendo buenos resultados los cuales se observan en la figura 5.10.

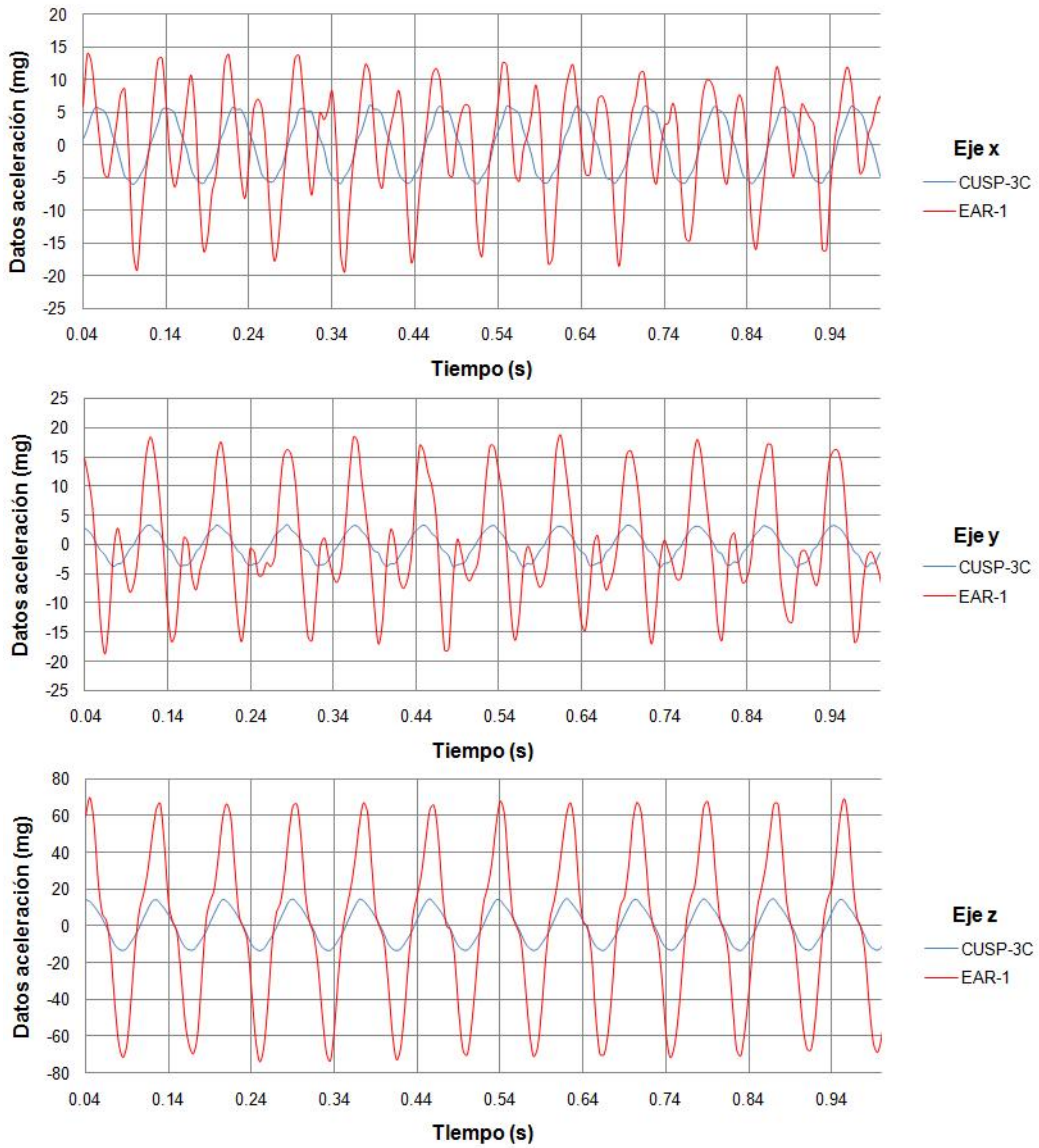
Una vez eliminado el rizado, se grafican los datos y se agrega una línea de tendencia, como se ejemplifica en las gráficas de la figura 5.11, en donde se observan los resultados obtenidos para cada eje.

De esta manera la ecuación de la recta que relaciona los datos de los dos equipos para cada uno de los ejes es:

Eje x:

$$y = 1{,}0241 \cdot x + 0{,}0165 \tag{5.2}$$

Figura 5.9. Datos con frecuencia de 12Hz para realizar calibración lineal.



Eje y:

$$y = 2{,}9519 \cdot x + 0{,}0746 \tag{5.3}$$

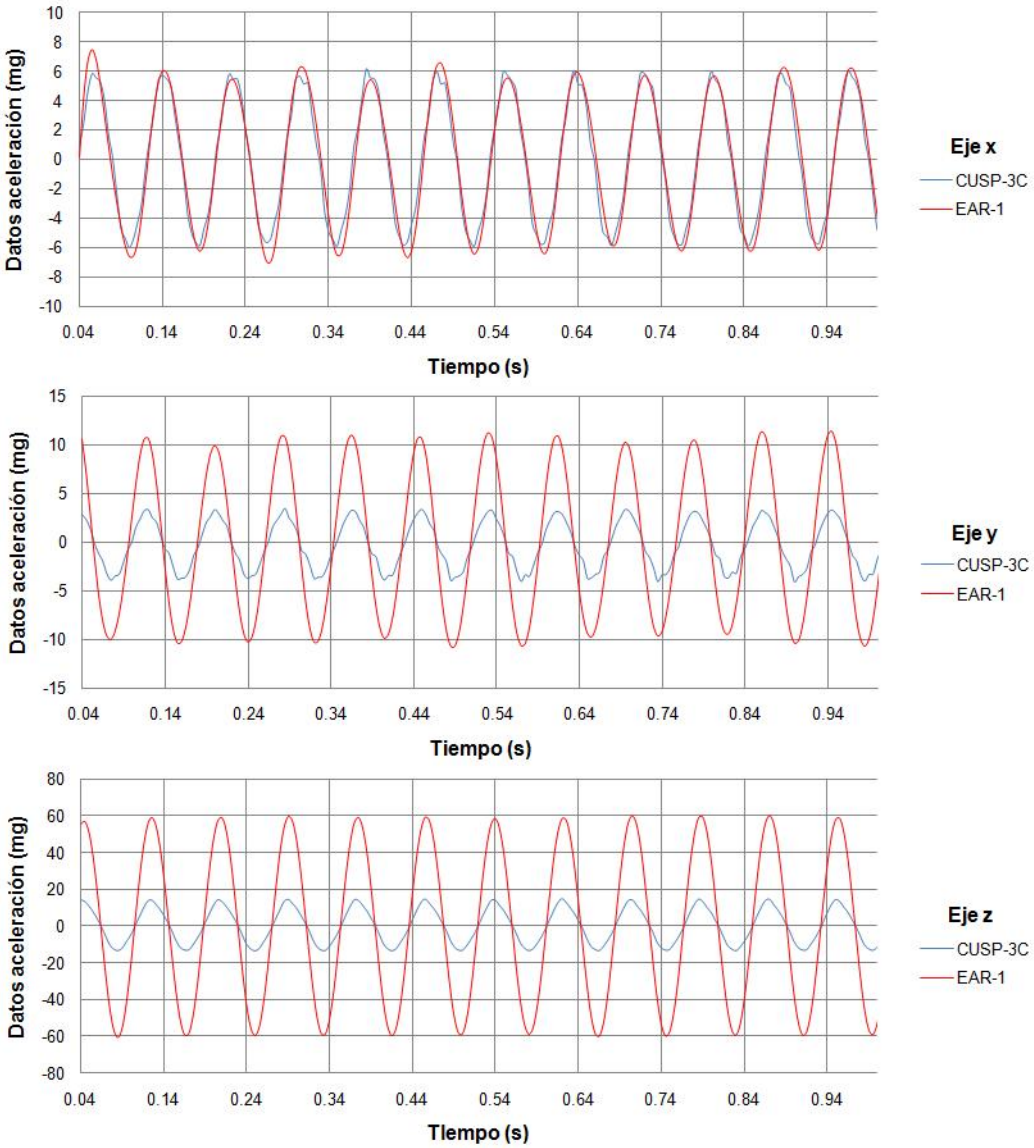
Eje z:

$$y = 4{,}6552 \cdot x + 0{,}0021 \tag{5.4}$$

como lo expresa Navarro[2], existe un parámetro que dice qué tan acertada fue la elección de la recta como curva de ajuste, el cual se denomina, *coeficiente de correlación* (R) y

[2]NAVARRO, Emilio. Prácticas de física. Valencia : Universidad Politécnica de Valencia - Servicio

Figura 5.10. Datos en 12Hz con filtro pasa-bajas aplicado.



toma valores entre 0 y 1. Cuanto mejor sea la aproximación de una recta más cercanos a 1 serán los valores del coeficiente R^2.

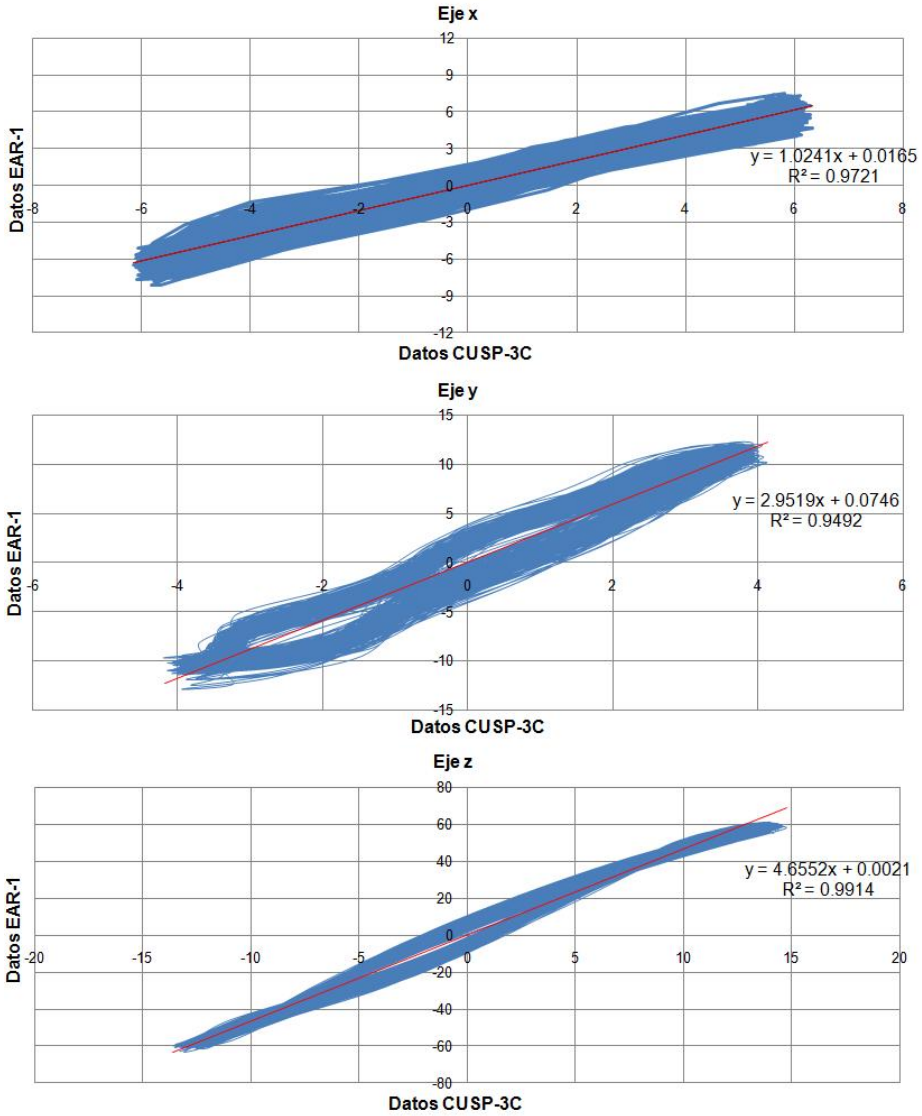
Los coeficientes de correlación para cada uno de los ejes son:

Eje x:

$$R^2 = 0{,}9721 \tag{5.5}$$

de Publicaciones, 2006. p. 129.

Figura 5.11. Regresión lineal de los datos con frecuencia de 12Hz.



Eje y:

$$R^2 = 0{,}9492 \tag{5.6}$$

Eje z:

$$R^2 = 0{,}9914 \tag{5.7}$$

Teniendo en cuenta las ecuaciones obtenidas en 5.2, 5.3 y 5.4, se calculan los nuevos datos del equipo EAR-1 y se comparan los resultados, los cuales se observan en las gráficas de la figura 5.12.

Luego se comparan los resultados encontrando el espectro de frecuencias para cada uno de los ejes de cada equipo, como se observa en las gráficas 5.13, 5.14 y 5.15 para valores de frecuencia de 6, 10 y 12Hz respectivamente.

Figura 5.12. Datos en el tiempo con frecuencia de 12Hz después de realizar la calibración lineal

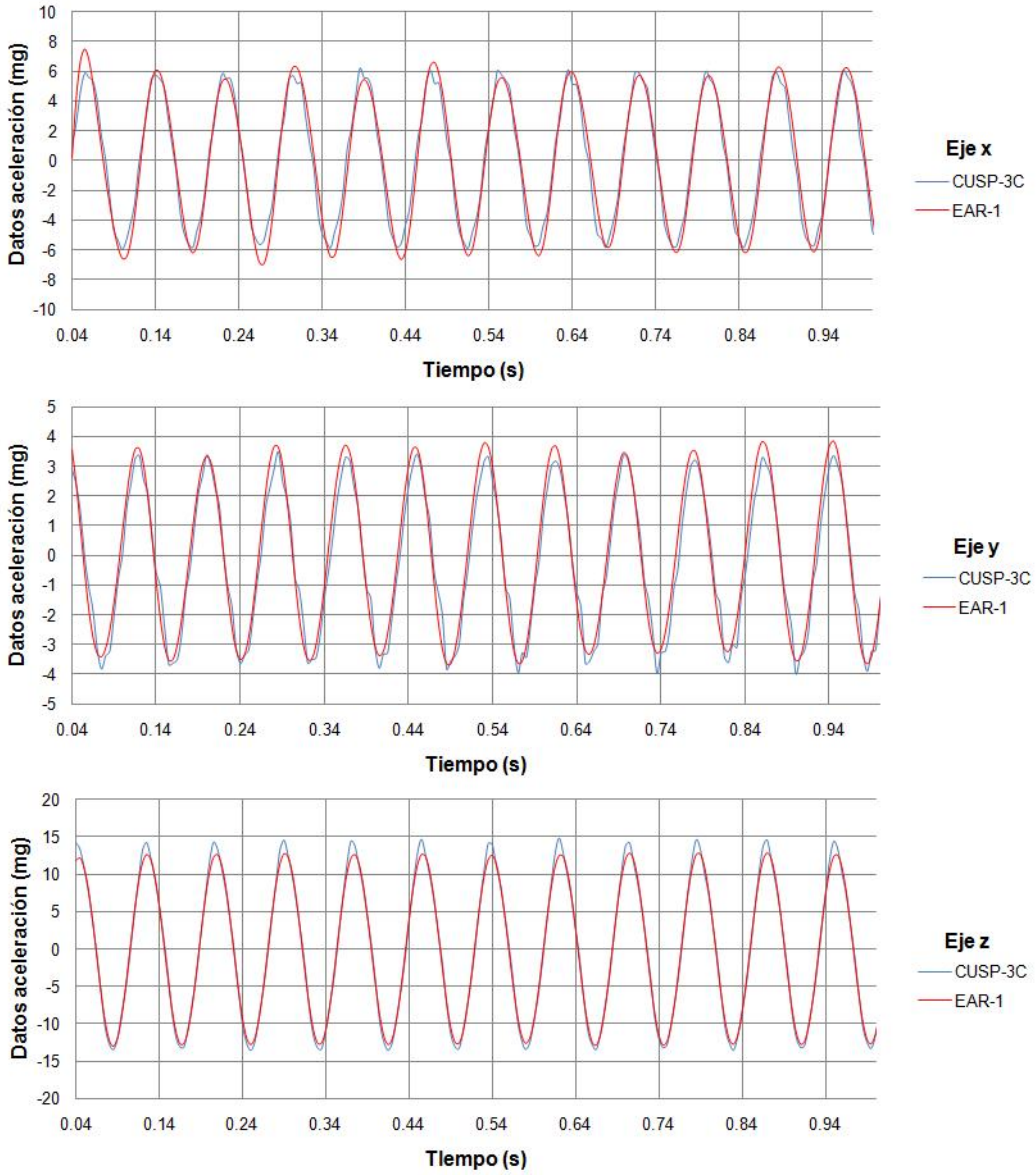
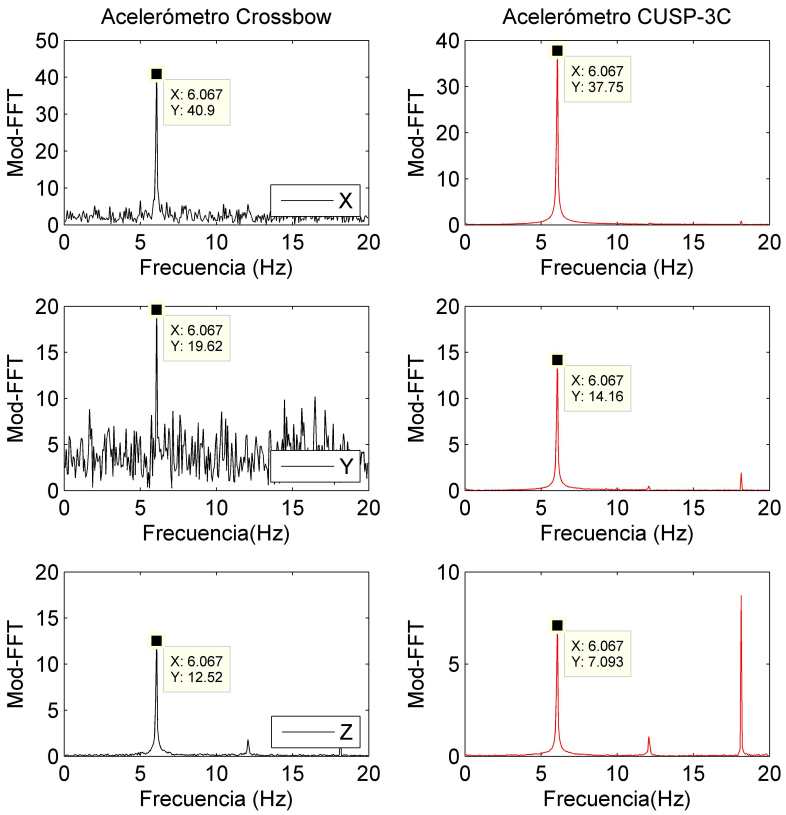


Figura 5.13. Resultados obtenidos en el dominio frecuencial para datos a 6Hz con calibración lineal.



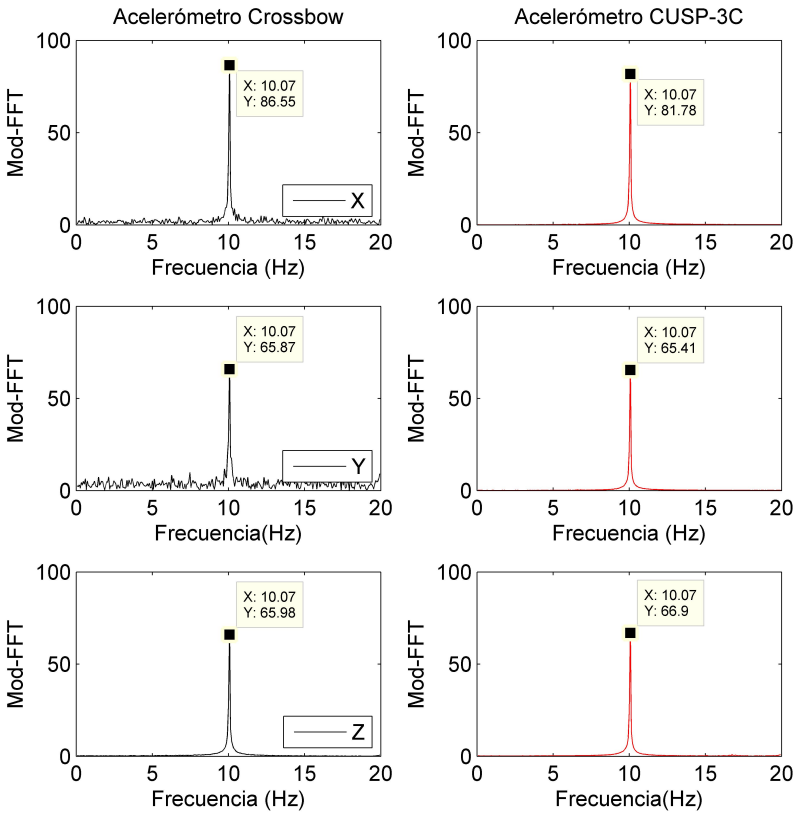
5.3.2. Identificación del sistema inverso. Pazos[3] propone el método de Pavlis y Vernon, para desarrollar una técnica de calibración empírica, por comparación de datos de ruido sísmico registrados por dos sensores suficientemente próximos, siendo conocida la respuesta de uno de ellos.

El espectro de un acelerograma, $s_i(t)$, registrado por el instrumento i, se relaciona con la aceleración del suelo, $e(t)$ mediante la ecuación 5.8, donde T_i es la respuesta en aceleración del acelerómetro i, A_i es la correspondiente del sistema A/D, y E es el movimiento (aceleración) real del suelo.

$$S(w) = T_i(w) \cdot A_i(w) \cdot E(w) \tag{5.8}$$

[3]PAZOS GARCÍA, Antonio. Estación sísmica digital tratamiento digital de señales. Cádíz : Universidad de Cádiz, 2003. p. 90-113.

Figura 5.14. Resultados obtenidos en el dominio frecuencial para datos a 10Hz con calibración lineal.



Si dos sensores se sitúan muy próximos, para poder suponer que el movimiento del suelo es el mismo para ambos, la función de transferencia entre ambos instrumentos es como la ecuación 5.9.

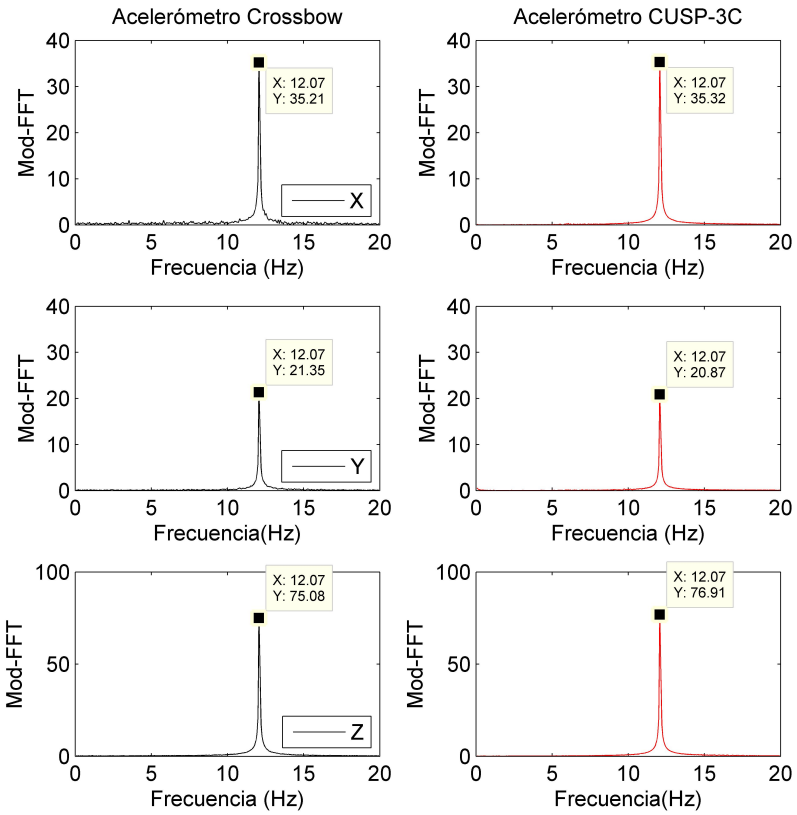
$$Z(w) = \frac{T_1(w)}{T_2(w)} = \frac{S_1(w) \cdot A_2(w)}{S_2(w) \cdot A_2(w)} \tag{5.9}$$

Si se conoce la respuesta del instrumento 2, se puede obtener la del otro por medio de la ecuación 5.10, que es independiente de la señal E que se utilice.

$$T_1(w) = Z(w) \cdot T_2(w) \tag{5.10}$$

Con el método de Pavlis y Vernon obtienen buenos resultados tomando como señal el ruido sísmico, y señalan los principales problemas para la estimación de la respuesta del instrumento que se desea calibrar, uno de ellos es la componente DC debida al efecto

Figura 5.15. Resultados obtenidos en el dominio frecuencial para datos a 12Hz con calibración lineal.



de los amplificadores, al conversor A/D y al acelerómetro. La forma más sencilla de eliminarla es simplemente restando la media del registro y eliminando las tendencias que pueda presentar la señal.

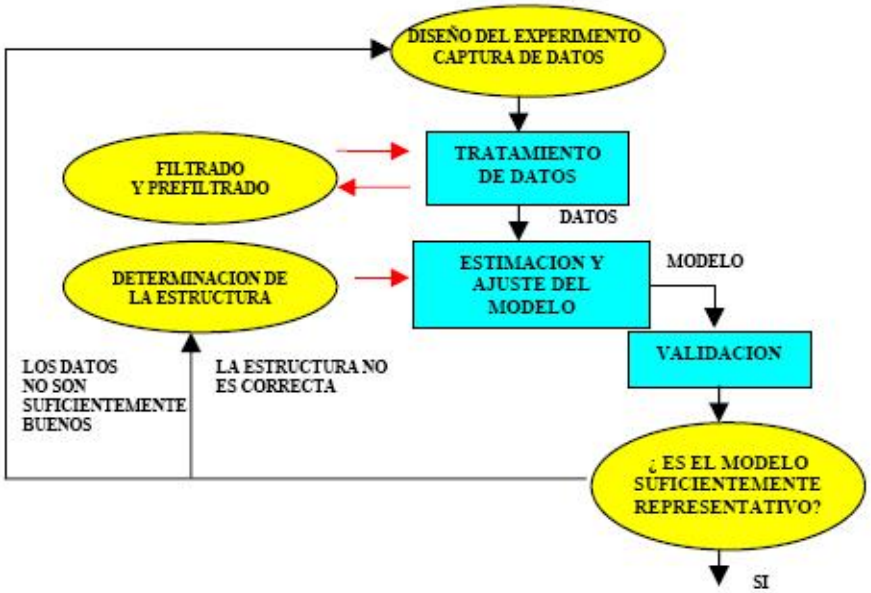
El segundo problema se refiere a la estimación de la respuesta entre ambos instrumentos (ecuación 5.9), una de las soluciones es recurrir al método de *identificación del sistema inverso*. Se denomina identificación a la técnica de construir un modelo matemático a partir de las variables medidas: entradas, salidas y, posiblemente, perturbaciones.

El procedimiento de identificación es iterativo, volviendo hacía atrás para determinar diferentes modelos y probar con diferentes estructuras. La figura 5.16 representa el proceso de identificación propuesto por Ljung.

Agudelo[4] reconoce que el software comercial más usado en la identificación de sistemas

[4]AGUDELO, Andrés. Estudio de la identificación de un sistema con el "system identifica-

Figura 5.16. Proceso de identificación propuesto por el profesor Lennart Ljung.



FRANCO, Javier; FLECHA, José; ALVAREZ, Alonso. Identificación de sistemas: Búsqueda de un modelo conceptual. Disponible en internet: `http://www.cea-ifac.es/actividades/jornadas/XXVIII/documentos/1221-XXVIIIJA%20CR3.pdf`. p. 2.

es el Toolbox de Mathworks "System Identification", desarrollado por Ljung y de uso con Matlab, agrupa los modelos de identificación más comunes y ayuda a predecir el comportamiento en datos de series temporales.

Los métodos matemáticos de identificación pueden ser paramétricos y no paramétricos. En los métodos paramétricos se selecciona un grupo restringido de posibles modelos y se busca encontrar los parámetros específicos de cada modelo. En los métodos no paramétricos no se busca explícitamente una lista de parámetros, se usan técnicas directas que no necesariamente requieren un conjunto confinado de posibles modelos.

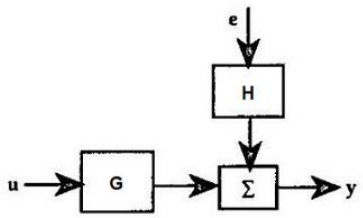
Estimación de modelos paramétricos. Se trata de construir modelos con base en el conocimiento experimental del comportamiento de las variables del sistema. Estos modelos pueden ser de tipo interno (espacio de estados) o de tipo externo (modelos de e/s), además se presupone con fines prácticos y sin perder demasiada precisión que el sistema es causal, lineal e invariante en el tiempo (LTI).

---

tion toolbox" de Matlab [en línea]. Disponible en internet: `http://www.google.com.co/#hl=es&source=hp&q=Estudio+de+la+identificaci%C3%B3n+de+un+sistema+con+el+%60%60system+identification+toolbox%27%27+de+Matlab.&btnG=Buscar+con+Google&meta=&aq=f&oq=Estudio+de+la+identificaci%C3%B3n+de+un+sistema+con+el+%60%60system+identification+toolbox%27%27+de+Matlab.&fp=b860d126d53f0aef`. p. 1.

Se debe tener en cuenta que en la práctica siempre hay señales fuera de control que afectan al sistema; en una estructura lineal se puede suponer que tales efectos pueden reunirse y sumarse a la salida del sistema, tal como se muestra en la figura 5.17.

Figura 5.17. Diagrama de un sistema LTI básico.



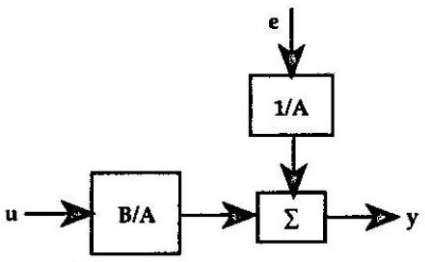
La descripción general de un modelo LTI está dada por la ecuación 5.11, donde G es un operador que captura la dinámica del sistema con los datos de entrada y salida, H es un operador que describe las propiedades de las perturbaciones aditivas a la salida y toma una entrada hipotética de fuente de ruido e, también llamado modelo de ruido. Cuando se estima un modelo de ruido, el toolbox incluye un canal e a la entrada de cada salida del sistema.

$$y = Gu + He \tag{5.11}$$

Para realizar la calibración se consideran los métodos paramétricos, dentro de los cuales se exploran los modelos lineales de caja negra ARX, ARMAX y de espacio estado. Estos modelos de estimación en el dominio temporal tienen la particularidad de que a partir de ellos se estiman modelos discretos (dominio-Z).

ARX (autorregresivo con variables exógenas). La representación de este modelo corresponde a la gráfica de la figura 5.18.

Figura 5.18. Representación modelo ARX.



AR se refiere a la parte regresiva $Aq^{-1}u(t)$ y X a la entrada adicional $Bq^{-1}u(t)$ (entrada exógena). Considera que la perturbación se somete de manera dinámica en el denominador similar al sistema, es decir, la parte determinista y estocástica tienen el mismo

denominador y su estructura se presenta en la ecuación 5.12, donde $n[k]$ representa el retardo del proceso y dentro del término $e(t)$ se consideran los errores de modelización y el ruido blanco.

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})}u(t - nk) + \frac{1}{A(q^{-1})}e(t) \tag{5.12}$$

$B(q)$ es un polinomio de grado nb que tiene la forma de la ecuación 5.13.

$$B(q^{-1}) = b_0 + b_1 q^{-1} + \cdots + b_{nb} q^{-nb} \tag{5.13}$$

El polinomio $A(q-1)$ es el polinomio autorregresivo de orden na como se expresa en la ecuación 5.14.

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na} \tag{5.14}$$

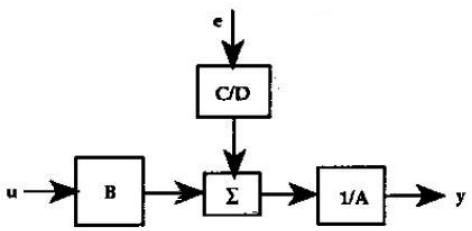
La desventaja de este modelo es la pérdida de una adecuada libertad para describir las propiedades del término perturbación.

ARMAX (autorregresivo de media móvil y variables exógenas) extiende el modelo ARX agregando ceros particulares a la perturbación ($C(q^{-1})$ como se ve en la ecuación 5.15. Tiene el mismo denominador y $C(q^{-1})$ es un polinomio parecido al de la ecuación 5.14 de orden *nc*.

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})}u(t - nk) + \frac{C(q^{-1})}{A(q^{-1})}e(t) \tag{5.15}$$

Este modelo se tipifica en la gráfica de la figura 5.19.

Figura 5.19. Representación modelo ARMAX.



Espacio de estados. El modelo de espacio de estados en tiempo discreto provee la misma forma de relación entre la entrada y la salida que el modelo ARX, la estructura es como en las ecuaciones 5.16 y 5.17, donde T es el intervalo de muestreo, $u(kT)$ es la entrada al instante de tiempo kT y $y(kT)$ es la salida al instante de tiempo kT.

$$x(kT + T) = Ax(kT) + Bu(kT) + Ke(kT) \tag{5.16}$$

$$y(kT) = Cx(kT) + Du(kT) + e(kT) \tag{5.17}$$

A partir de los métodos mencionados, mediante el toolbox de Matlab, es posible entonces obtener de manera separada, tanto el modelo dinámico del sistema como el del ruido.

Experimentación y resultados. Se ubica el acelerógrafo CUSP-3C (instrumento patrón) y el acelerómetro Crossbow del equipo EAR-1 a una distancia de pocos centímetros, orientando el eje x hacia el norte cardinal y cuidando la nivelación de ambos.

Cada evento registrado dura un lapso de tiempo igual a 10 minutos utilizando una frecuencia de muestreo de 200Hz. Con estos registros se trata de seleccionar de manera apropiada la estructura del modelo para una exitosa aplicación de identificación. Con cada una de las aproximaciones encontradas se realizan los siguientes pasos:

- Se seleccionan los valores de los ejes x, y y z correspondientes a los dos instrumentos, teniendo como entrada las variables del equipo EAR-1 y como salida las obtenidas con el instrumento patrón.
- Se convierten los valores de voltaje, obtenidos con el acelerómetro Crossbow, a aceleración.
- Se realiza corrección de línea de base.
- Se filtran los datos con un filtro butterworth de octavo orden con frecuencia de corte en 20Hz.
- Los datos de cada uno de los ejes se introducen en el toolbox de identificación de sistemas de Matlab.
- Se escogen los mejores modelos para cada uno de los ejes y se comprueba su estabilidad.
- Con cada modelo se obtiene una salida estimada a partir de los datos del equipo EAR-1 y se grafica en el dominio del tiempo y de la frecuencia para observar cual es la mejor aproximación.

Se escoge uno de los eventos que se estima proporcionan una correcta información y para cada una de las componentes (x, y y z), se prueba con los diferentes modelos, los cuales arrojan resultados de ajuste entre el 98 % y el 99 %.

Los mejores resultados se obtienen con $na = 5$ (número de polos), $nb = 5$ (número de ceros) y retardo $nk = 1$. En el cuadro 5.2 se indica el porcentaje de ajuste para los modelos encontrados que más se aproximan en la estimación.

No es posible obtener una estimación con los modelos ARMAX cambiando el orden de nc (perturbación de entrada) a un valor mayor de cero, puesto que se vuelve inestable, es por eso que el resultado es igual al del modelo ARX en todos los casos.

Cuadro 5.2. Ajuste de los modelos encontrados para la identificación del sistema.

Eje	Método	Ajuste
x	*Espacio de estados*	
	Método subespacio(N4SID) orden 5	98.89 %
	Método predicción de error(PEM) orden 5	99.33 %
	ARX na = 5, nb = 5, nk = 1	99.33 %
	$ARMAX$ na = 5, nb = 5, nc = 0, nk = 1	99.33 %
y	*Espacio de estados*	
	Método subespacio(N4SID) orden 5	98.88 %
	Método predicción de error(PEM) orden 5	99.31 %
	ARX na = 5, nb = 5, nk = 1	99.36 %
	$ARMAX$ na = 5, nb = 5, nc = 0, nk = 1	99.36 %
z	*Espacio de estados*	
	Método subespacio(N4SID) orden 5	98.84 %
	Método predicción de error(PEM) orden 5	99.29 %
	ARX na = 5, nb = 5, nk = 1	99.18 %
	$ARMAX$ na = 5, nb = 5, nc = 0, nk = 1	99.18 %

Para todos los modelos no se aumenta el orden de polos o de ceros a un número mayor de cinco, puesto que se vuelven inestables, además si se aumenta nk (retardo) para los modelos ARX y ARMAX disminuye la estimación, lo contrario sucede en espacio de estados donde se mejora el ajuste, aunque en valores muy cercanos.

De los dos métodos lineales de espacio de estados, N4SID se usa para estimar un modelo inicial que se mejora con el método iterativo de predicción de error (PEM), el cual es mas preciso y robusto.

Mediante el comando *tf* en Matlab se obtienen la funciones de transferencia a partir de los polinomios o ecuaciones de estado según sea el caso, del modelo dinámico del sistema (G) y del modelo de ruido (H), teniendo en cuenta la ecuación general de un sistema lineal ecuación 5.11. Para evaluar la influencia de las perturbaciones se genera un ruido blanco gaussiano aleatorio de media cero y con varianza igual al parámetro *NoiseVariance* usado por cada método.

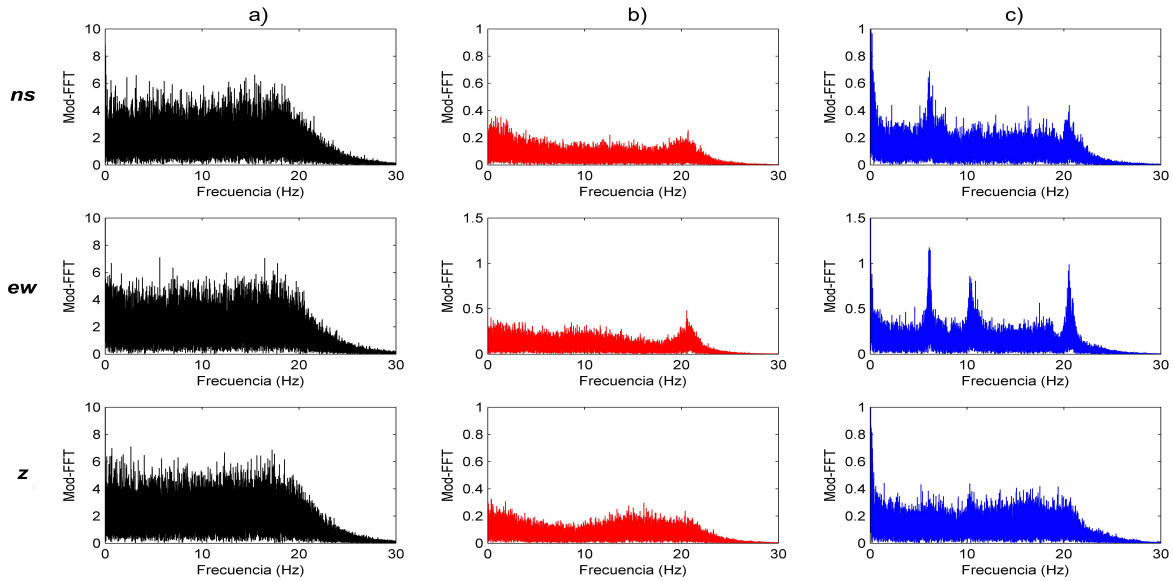
El cálculo de los datos estimados para cada una de las componentes se ejemplifica en la siguiente rutina de matlab, esto se realiza para cada modelo probado.

```
nvx = 5.3868e-008; %NoiseVariance para x
nvy = 5.7120e-008; %NoiseVariance para y
nvz = 1.3269e-007; %NoiseVariance para z

%Ruido gaussiano
noisez = random('norm', 0, sqrt(nvz),size(zf),1);
noisey = random('norm', 0, sqrt(nvy),size(yf),1);
noisex = random('norm', 0, sqrt(nvx),size(xf),1);

%Calculo de las componentes estimadas
%Componente x
xn = filter(numx,denx,xf) + filter(nnx,dnx,noisex);
%Componente y
yn = filter(numy,deny,yf) + filter(nny,dny,noisey);
%Componente z
```

```
zn = filter(numz,denz,zf) + filter(nnz,dnz,noisez);
```

En la figura 5.20 se muestra la respuesta obtenida al utilizar el método PEM de espacio de estados para los datos registrados con el equipo EAR-1 de los ejes x, y y z. En la primera columna se grafican los datos sin corregir, en la segunda columna los datos resultado de la calibración aplicando el modelo y en la tercera columna la respuesta del instrumento patrón.

Figura 5.20. Espectros de frecuencia para datos calibrados con método PEM: a) Datos EAR-1 sin calibrar, b) Datos EAR-1 calibrados y c) Datos CUSP-3C.

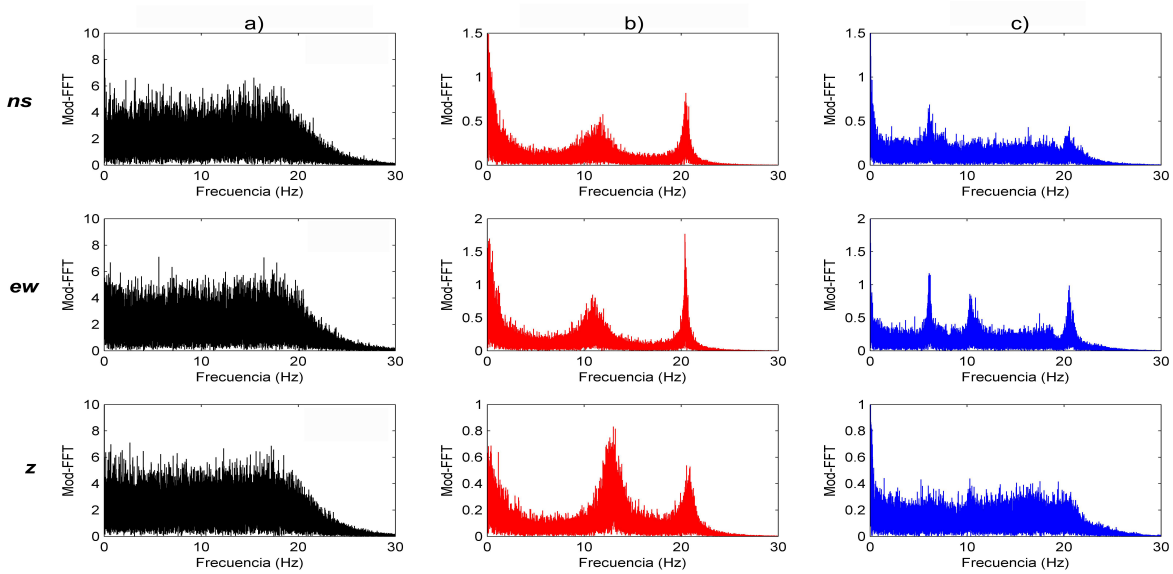


Se grafica en el dominio de la frecuencia para poder observar los espectros de Fourier y así comparar la similitud entre los dos instrumentos en cuanto a la captación de señales a frecuencias bajas, teniendo en cuenta como parámetros principales los valores de amplitud y de frecuencia que son la herramienta principal para aplicar el método de Nakamura.

En la figura 5.21 se grafican los espectros de frecuencia obtenidos para la calibración por medio de los modelos ARX obtenidos.

El modelo que brinda una mejor aproximación es el de espacio de estados con el método de predicción de error, por lo tanto, es el modelo escogido para corregir los datos obtenidos con el equipo EAR-1.

Figura 5.21. Espectros de frecuencia para datos calibrados con método ARX: a) Datos EAR-1 sin calibrar, b) Datos EAR-1 calibrados y c) Datos CUSP-3C.



Se realizaron varias pruebas comparativas en diferentes lugares de la ciudad de San Juan de Pasto como se describe a continuación.

1. Evento No. 1: Barrio La Castellana

Se registra durante un lapso de 10 minutos las señales captadas por el acelerógrafo CUSP-3C y el equipo EAR-1 en el Barrio La Castellana, en la figura 5.22 se muestran los datos adquiridos por los dos instrumentos.

Al aplicar el modelo obtenido a cada una de las componentes, se estiman los valores del EAR-1 con respecto al acelerógrafo CUSP-3C. En la figura 5.23 se observan las gráficas en el dominio de la frecuencia para los datos originales, los datos después de la estimación con el modelo PEM de espacio de estados y los del instrumento patrón.

2. Evento No. 2: Barrio La Primavera

De las pruebas realizadas en el barrio La Primavera se obtiene un registro de 10 minutos, del cual se muestra cada una de las componentes adquiridas por los dos instrumentos en la figura 5.24.

Figura 5.22. Datos de registro del evento No. 1: a) Acelerógrafo CUSP-3C y b) Equipo EAR-1.

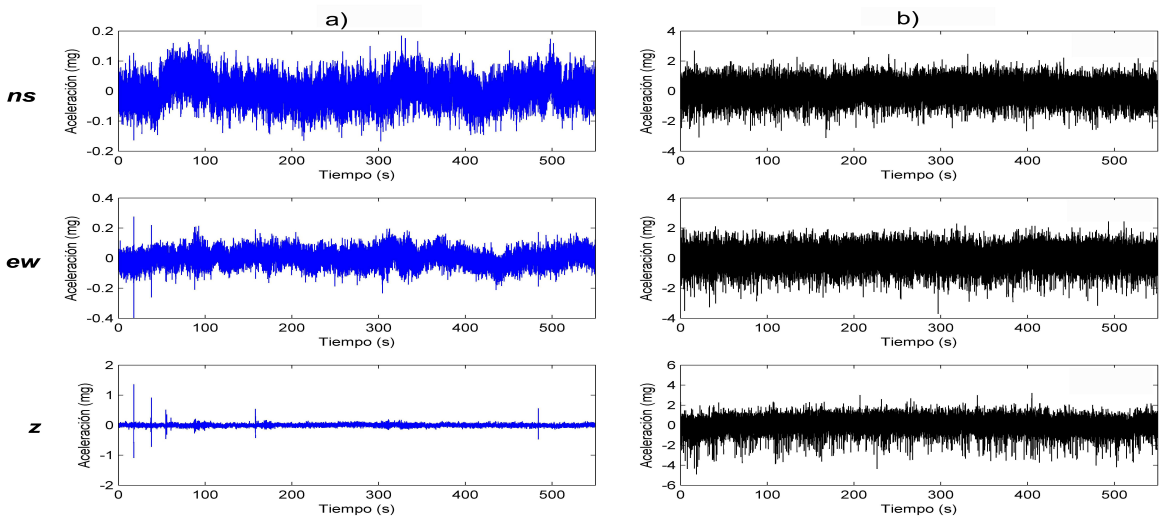


Figura 5.23. Espectros de frecuencia para datos del evento No. 1: a) Datos EAR-1 sin calibrar, b) Datos EAR-1 calibrados y c) Datos CUSP-3C.

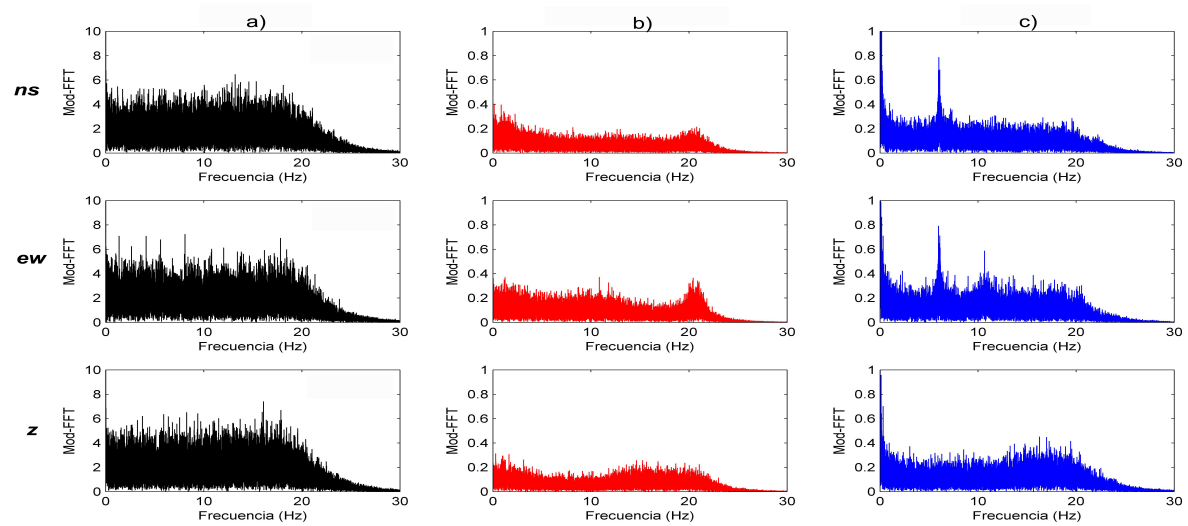
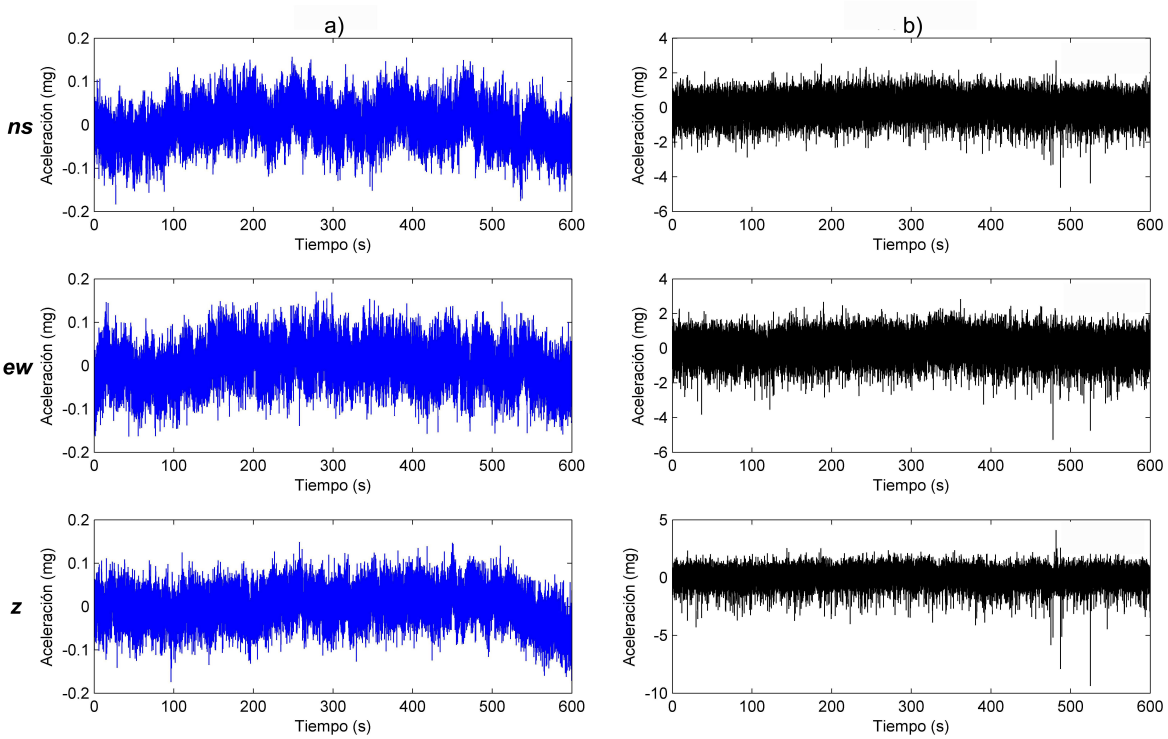


Figura 5.24. Datos de registro del evento No. 2: a) Acelerógrafo CUSP-3C y b) Equipo EAR-1.



Al aplicar el modelo pem se obtienen datos estimados a partir de datos los registrados con el equipo EAR-1. Se grafica en el dominio de la frecuencia como se muestra en la figura 5.25.

3. Evento No. 3: Barrio San Miguel

De las pruebas realizadas en el barrio San Miguel se escoge un registro de un lapso de 5 minutos. Las gráficas de aceleración con respecto al tiempo de los datos obtenidos se indican en la figura 5.26.

Se aplica el modelo pem y se encuentra la estimación para los datos del equipo EAR-1, los datos obtenidos se grafican en el dominio de la frecuencia como se muestra en la figura 5.27.

Figura 5.25. Espectros de frecuencia para datos del evento No. 2: a) Datos EAR-1 sin calibrar, b) Datos EAR-1 calibrados y c) Datos CUSP-3C.

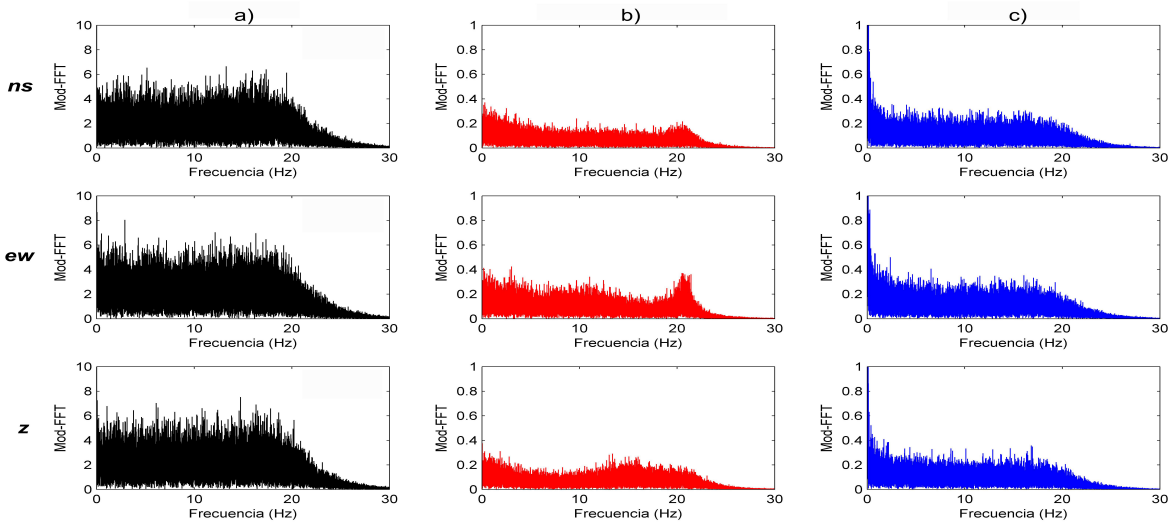


Figura 5.26. Datos de registro del evento No. 3: a) Acelerógrafo CUSP-3C y b) Equipo EAR-1.

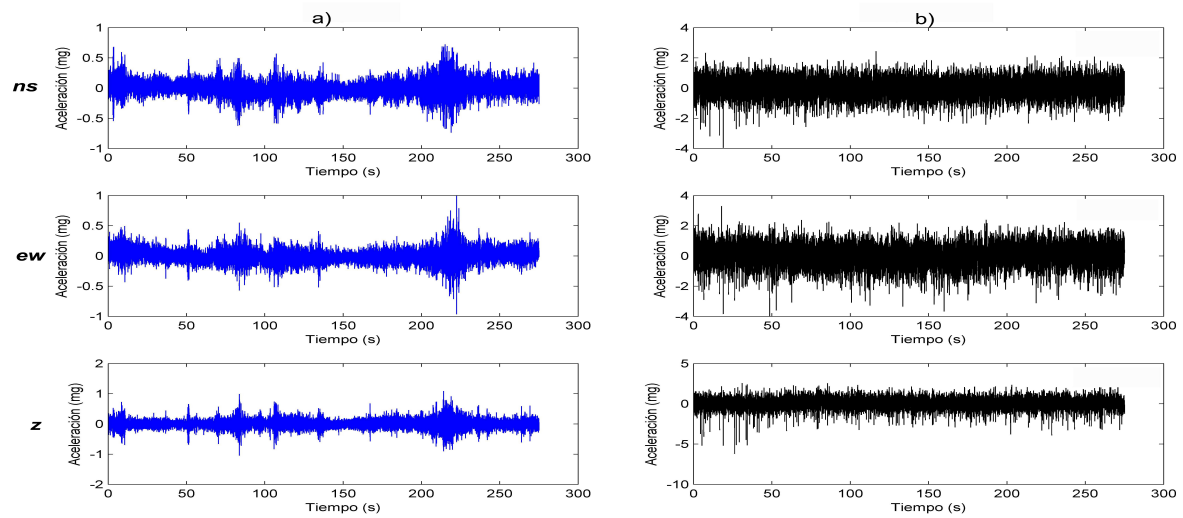
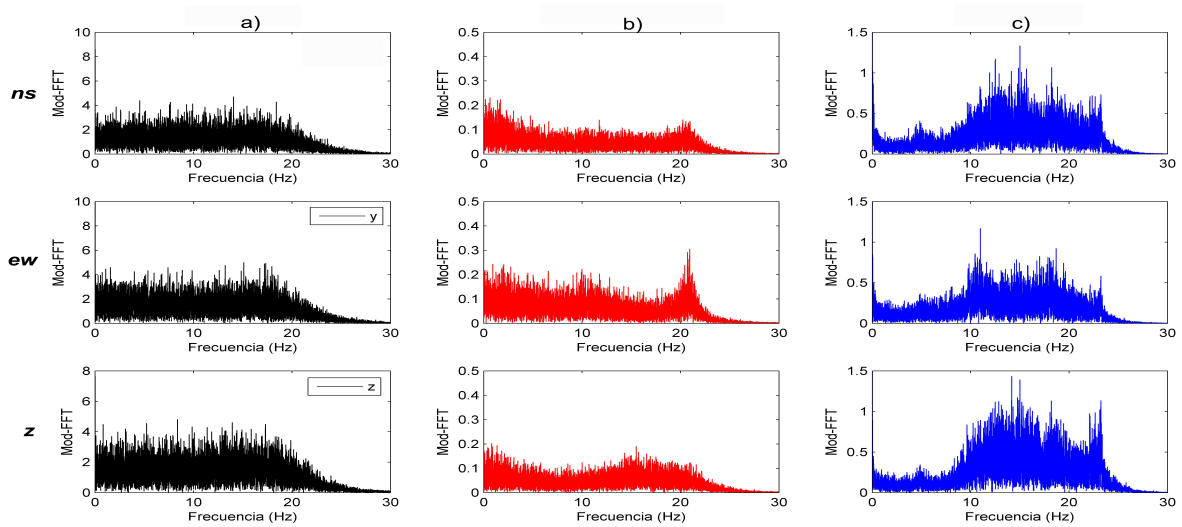


Figura 5.27. Espectros de frecuencia para datos del evento No. 3: a) Datos EAR-1 sin calibrar, b) Datos EAR-1 calibrados y c) Datos CUSP-3C.



4. Evento No. 4: Universidad de Nariño

En las instalaciones del laboratorio de la Universidad de Nariño se realiza una prueba en la que se obtiene un registro de 10 minutos. En las graficas de la figure 5.28 se observan los datos de aceleración en el tiempo de los dos instrumentos, para las tres componentes.

Al aplicar el modelo pem se obtienen los datos estimados a partir de los registrados por el equipo EAR-1, en la figura 5.29 se observan los espectros de frecuencia para las datos de los dos instrumentos utilizados y los estimados.

Figura 5.28. Datos de registro del evento No. 4: a) Acelerógrafo CUSP-3C y b) Equipo EAR-1.

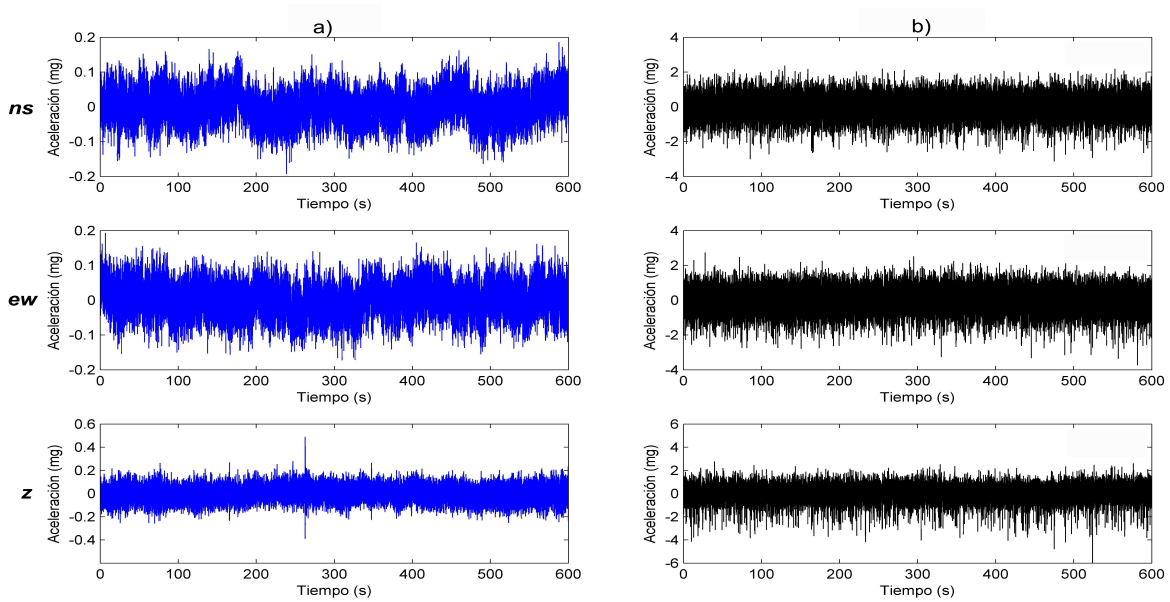
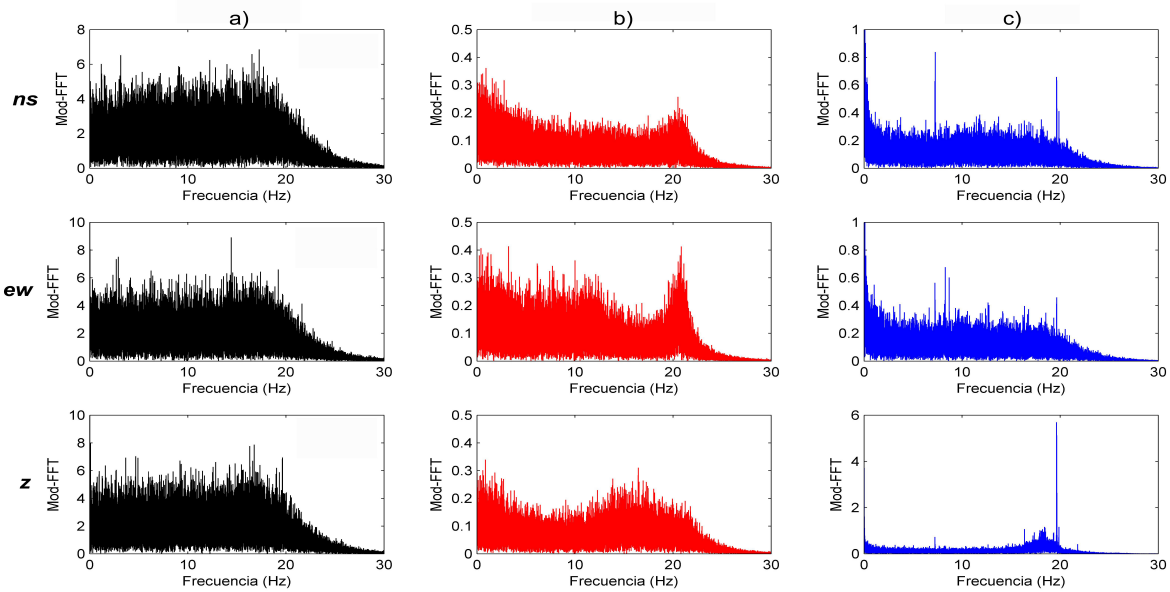


Figura 5.29. Espectros de frecuencia para datos del evento No. 4: a) Datos EAR-1 sin calibrar, b) Datos EAR-1 calibrados y c) Datos CUSP-3C.



6 CONCLUSIONES

Con el diseño y la implementación del equipo de adquisición y registro de señales microsísmicas (EAR-1) es posible captar vibraciones de baja frecuencia traducidas en aceleración.

Los componentes utilizados para el equipo EAR-1 se basan en los requisitos presentados en la norma NSR-98, Título A, Capítulo A-11, en donde se estipulan las características para la instalación de acelerógrafos.

Mediante la etapa de acondicionamiento implementada basada en amplificadores y filtros, se garantiza la adecuación de las señales provenientes del acelerómetro triaxial utilizado dentro del rango de frecuencias de interés, definido desde DC a 20Hz.

En el equipo EAR-1 se garantiza una adquisición simultánea entre las señales de las componentes norte-sur, este-oeste y vertical del acelerómetro, gracias a que la tarjeta de adquisición de datos NI USB-9215 posee está característica entre sus canales.

El software de interfaz de usuario desarrollado es una herramienta eficaz que cumple con unas características previamente definidas para facilitar el control de los parámetros en la adquisición y registro de señales y el tratamiento de los datos registrados, esto es posible debido a la integración realizada entre Matlab y Visual C++.NET.

Matlab es una plataforma versátil que permite realizar de una forma sencilla el procesamiento digital de datos, además, posibilita la creación de un entorno gráfico mediante la herramienta Matlab Guide.

Para evitar limitaciones en el uso del software es importante la creación del mismo como una aplicación independiente de la plataforma usada para su desarrollo y que requiera condiciones mínimas para su ejecución.

Por medio de pruebas comparativas entre el equipo EAR-1 y el acelerógrafo CUSP-3C del Centro de Investigación en Ingeniería Sismológica, es posible realizar una serie de ajustes a los datos, esto permite corregir errores ocasionados por los componentes electrónicos utilizados, perturbaciones y demás distorsiones que afectan la señal.

Es importante tener en cuenta que la señal digital resultante de la conversión analógico-digital realizada en la etapa de adquisición del equipo EAR-1, es sensiblemente diferente a la señal de entrada que la origina, debido a esto, siempre va a existir una cierta diferencia entre ambas que depende de la resolución del conversor utilizado. El acelerógrafo CUSP-3C presenta una resolución de 24 bits, que reproduce la señal de manera más exacta en comparación al equipo EAR-1 con resolución de 16 bits.

7 RECOMENDACIONES

A partir de los resultados obtenidos en el desarrollo del equipo EAR-1 se realizan las siguientes recomendaciones:

- Implementar un reloj interno de alta presición calibrable o un sistema de posicionamiento global (GPS) como lo acordado en la norma NSR-98, Título A, Capítulo A-11: Requisitos mínimos para la instalación de acelerógrafos.
- Incorporar un medio de almacenamiento magnético para el registro de las señales.
- Incluir nuevas herramientas en el software para el tratamiento de señales.
- Profundizar en otros métodos de calibración para realizar una optimización en la corrección instrumental del equipo.
- Utilizar el software desarrollado para otro tipo de aplicaciones que requieran adquirir y registrar señales análogas.
- Diseñar e implementar un equipo que permita el registro de sismos fuertes, el cual puede basarse en el principio de funcionamiento del EAR-1, haciendo las modificaciones correspondientes en cuanto al acondicionamiento de la señal.

BIBLIOGRAFÍA

AGUDELO, Andrés. Estudio de la identificación de un sistema con el "system identification toolbox" de Matlab [en línea]. Disponible en internet: http://www.google.com.co/#hl=es&source=hp&q=Estudio+de+la+identificación%C3%B3n+de+un+sistema+e+%60%60system+identifica+toolbox%27+de+Matlab.&bt&meta=&aq=f&oq=Estud+d+la+identificaci%C3%B3n+de+un+sistema+\con+el+%60%60system+identifica+tool%27%27+de+Matlab.&fp=f0aef. p. 1.

ALFARO, A.; NAVARRO, J.; SÁNCHES, J. Microzonificación sísmica de Barcelona utilizando el método de Nakamura Ventajas y Limitaciones. Barcelona: Universidad Politécnica de Cataluña. Departamento de Ingeniería del Terreno y Cartográfica. (1999) p. 1.

ALFARO, A. et al. Determinación de características dinámicas del suelo a partir de microtemblores [en línea]. Disponible en internet: http://andresalfaro.com/doc_estudios/1999%20_alfaro_egozcue_ugalde.pdf. p. 1-3.

BAKER, Bonnie. Anti-aliasing, analog filters for data acquisition systems. Disponible en internet: http://ww1.microchip.com/downloads/en/AppNotes/00699b.pdf. p. 4-5.

BALCELLS, Josep; DAURA, Francesc y otros. Interferencias electromagnéticas en sistemas electrónicos. Barcelona : Marcombo, 1991. p. 85.

BERMUDEZ, María Luisa et al. Uso de las microtrepidaciones para la evaluación de la respuesta dinámica de los suelos. [en línea], 2002. Disponible en internet: http://bdrsnc.ingeominas.gov.co/publicaRNAC/PUBLICACIONES/SISMOLOGIA_2002/MICRO_RESPUESTA_DINAMICA_DE_SUELOS.PDF.

BURR-BROWN, INA128 precision, low power instrumentation amplifiers [en línea]. U.S.A., 1996. Disponible en internet: http://www.datasheetcatalog.org/datasheet/texasinstruments/ina128.pdf. p. 8.

CARACENA, Teófilo. Instrumentación Industrial: Acelerómetros [en línea]. Disponible en Internet: http://74.125.47.132/search?q=cache:B9OrwglNRPQJ:www.gii.upv.es/personal/gbenet/IIN/treballs%25200607%C3+Industrial:+Acelerometros.&\cd=1&hl=es&ct=clnk&gl=co. p. 5.

CARREÑO E., et al. Registro y Tratamiento de Acelerogramas. En : Física de la Tierra. Vol. 1 (1989) p. 81-107.

CORAL MONCAYO, Hugo. Ingeniería Sismológica. Pasto: Universidad de Nariño. Departamento de Ingeniería Civil (2009) p. 114-116, 378-380.

CROSSBOW TECHNOLOGY, CXL-GP Series Data Sheet [en línea]. Disponible en Internet: http://www.hoskin.qc.ca/uploadpdf/Instrumentation/CrossBow/hoskin_GP%20Series_47e00ece030de.pdf. p. 2.

CROSSBOW TECHNOLOGY, CXL-HF Series. Vibration monitoring accelerometers [en línea]. Disponible en internet: http://www.instrumentation.it/main/pdf/cross\bow/HF_Series_ID.pdf. p. 2.

FRANCO, Javier; FLECHA, José; ALVAREZ, Alonso. Identificación de sistemas: Búsqueda de un modelo conceptual. Disponible en internet: http://www.cea-ifac.es/activi\dades/jornadas/XXVIII/documentos/1221-XXVIIIJA%20CR3.pdf. p. 2.

FERNÁNDEZ OLTRA, Rubén. Sistema de Adquisición de Posicionamiento Geográfico [en línea]. Disponible en Internet: http://upcommons.upc.edu/pfc/bitstream/2099.1/4453/1/Sistema%20de%20Adquisici%C3%B3n%20de%20Posicionamiento%20Geogr%C3%A1fico.pdf. p. 18-26.

Glosario Geológico Minero. [en línea]. Versión 1.8. Bogotá. Instituto Colombiano de Geología y minería, 2005. Disponible en Internet: http://www.ingeominas.gov.co/index.php?option=com_glossary&catid=82&func=\display&search=sismo.

GÓMEZ, Marisol; ZORRILLA, Juan Carlos; MONSALVE, Jaramillo. La electrónica como soporte instrumental de la sismología. Observatorio Sismológico de la Universidad del Quindío. (2009) p. 2.

HAYS, Walter. Aspectos fundamentales de la geología y la sismología para la microzonificación sísmica. En : Física de la Tierra. Vol. 1 (1989); p. 217-250.

HECKERT A., Paul. Understanding the faraday cage. Disponible en internet: http://electricitymagnetism.suite101.com/article.cfm/understanding_the_faraday_cage.

Instrumentación Sísmica. [en línea]. Mexico. Centro Nacional de Prevención de Desastres CENAPRED. Disponible en Internet: http://cidbimena.desastres.hn/docum/crid/Abril2004/pdf/spa/doc2249/doc2249-d.pdf.

INTERSIL, ICL7660 Data Sheet [en línea]. Disponible en internet: www.intersil.com/data/fn/fn3072.pdf.

MAGGIOLO, Gustavo. Nota de Aplicación: detector sísmico con acelerómetro XYZ MMA7260Q. UTN - Facultad Regional Paraná (Argentina). Rev.N, 006/2007. Disponible en internet: Disponible en internet: http://www.electrocomponentes.com.ar/educa\cion/pdf/programa/2007/resumenes/UTN_PARANA_DETECTOR_SISMICO.pdf. p. 3.

MATHWORKS. Matlab Introduction and Key Features. Disponible en internet: http:

//www.mathworks.com/products/matlab/description1.html?s_cid=ML.

MATHWORKS. Matlab Documentation System Identification Toolbox: detrend [en línea], 2009. Disponible en internet: http://www.mathworks.com/access/helpdesk/help/toolbox/ident/index.html/access/helpdesk/help/toolbox/ident/detrend.html&http://www.mathworks.com/support/functions/alpha_list.html?sec=2.

MICROSOFT CORPORATION. Microsoft Developer Network (MSDN) Library [en línea]. Disponible en internet: http://msdn.microsoft.com/en-us/library/default.aspx.

MORALES, Johnny. Introducción al Procesamiento de Señales y al Análisis de Sistemas de Variable Discreta [en línea]. Disponible en Internet: http://electronica.udea.edu.co/cursos/circuitos3/20071/Pres1.pdf.

NATIONAL INSTRUMENTS CORPORATION, NI-DAQmx Base 2.x C Function Reference Help [en línea]. 1°edición, ago. 2006. Disponible en internet: http://student.agh.edu.pl/~koskak/ksp2/docsource/daqmxbasecfuncindex.htm.

NATIONAL INSTRUMENTS CORPORATION, Operating Instructions USB-9215 [en línea], 2004. Disponible en internet: ftp://ftp.ni.com/support/manuals/371261a.pdf. p. 1-12.

NATIONAL INSTRUMENTS CORPORATION. Tutorial: Acondicionamiento de Señales [en línea]. Disponible en Internet: http://digital.ni.com/worldwide/latam.nsf/87e62f4c89ea9df9862564250075e6e4/a6c5283/FILE/Acondicionamiento%20de%20%C3%B1ales.pdf. p. 4.

NAVARRO, Emilio. Prácticas de física. Valencia : Universidad Politécnica de Valencia - Servicio de Publicaciones, 2006. p. 129.

PALLÁS, Ramón. Adquisición y distribución de señales. Barcelona : Marcombo, 1993. p. 255-277.

PAZOS GARCÍA, Antonio. Estación sísmica digital tratamiento digital de señales. Cádíz : Universidad de Cádiz, 2003. p. 90-113.

PERNIA, Daniel. Introducción a la medición de vibración. Mérida, Venezuela: s.n. 2004, p. 2-6.

RIU, Jordi; BOQUÉ, Ricard. Calibración Lineal [en línea]. Disponible en internet: http://www.quimica.urv.es/quimio/general/callin.pdf. p. 1-3.

ROCA, Antoni. Instrumentación para campo cercano y análisis de acelerogramas. En : Física de la Tierra. Vol. 1 (1989) p. 133.

ROMBERG, Justin. Nyquist Theorem [en línea]. Disponible en internet: http://cnx.org/content/m10791/2.4

ROMO PROAÑO, Marcelo. Técnica para la generación de diagramas de velocidades y desplazamientos a partir de acelerogramas sísmicos. Centro de Investigaciones científicas, Escuela politécnica del ejército. p. 8.

ROSSO, Ana. Estimación del area bajo la curva utilizando una planilla de cálculo [en línea] Disponible en internet: www.union-matematica.org.ar. p. 1-3.

RUBIA, Juan. Nociones básicas sobre adquisición de señales [en línea]. Versión 1, diciembre de 2000. Disponible en Internet: http://www.redeya.com/electronica/tutoriales/adatos/adquisicion.html.

Tema 3. Adquisición de datos [en línea]. Disponible en Internet: http://www.depeca.uah.es/wwwnueva/docencia/IT-INF/ctr-eco/Tema3.pdf. p. 1.

THALES: SOCIEDAD ANDALUZA DE EDUCACIÓN MATEMÁTICA. Leyes de Newton [en línea]. Disponible en Internet: http://thales.cica.es/rd/Recursos/rd98/Fisica/02/leyes.html.

TORRES M., Gilbert Francisco. Importancia de la microzonificación sísmica de las principales ciudades del estado de Veracruz [en línea]. Disponible en Internet: http://www.smsp.org.mx/rhigiene/docs/Importancia%20de%20la%20Microzonificaci%C3%B3%20%28Torres%20Morales.%29.doc.

VALLVERDÚ, Francesc, RODRÍGUEZ, José A. y MORENO, Asunción. Tratamiento digital de la señal - Una introducción experimental. Barcelona : s.n., 1995. p. 17-152, 213-252.

XAVIEN, "X-3AX-ACC-EM" - Xavien 3AXis Accelerometer Engineering Module User Manual and Instructions. Arizona (USA). Disponible en Internet: http://www.geocities.com/researchtriangle/lab/6584/X_3AX_ACC_EM_MANUAL.pdf.

ANEXO A. Código interfaz de usuario Adquisición y Registro (Visual C++)

```
/********************** PROGRAMA DE ADQUISICIÓN Y REGISTRO ******************************
Este programa permite controlar los parámetros de la tarjeta de adquisición de datos
NI USB-9215 de la National Instruments, configurada por defecto para adquirir en modo
continuo durante un período de tiempo determinado.
================================================================================
                      UNIVERSIDAD DE NARIÑO
                      FACULTAD DE INGENIERÍA
               CENTRO DE INVESTIGACIÓN EN INGENIERÍA
                        SEPTIEMBRE 2009
================================================================================
********************************************************************************/

#pragma once
#include "ClaseAdquisición.h"
#include "PanelBienvenida.h"
#include "PanelInfoEvento.h"

namespace InterfazAdquisición {
   using namespace System;
   using namespace System::ComponentModel;
   using namespace System::Collections;
   using namespace System::Windows::Forms;
   using namespace System::Data;
   using namespace System::Drawing;
   using namespace System::IO;
   using namespace System::Timers;
   using namespace System::Globalization;
   using namespace System::Diagnostics;
   using namespace System::Windows;
   using namespace ZedGraph;
   /// <summary>
   /// Resumen de Form1
   ///
   /// ADVERTENCIA: si cambia el nombre de esta clase, deberá cambiar la
   ///          propiedad 'Nombre de archivos de recursos' de la herramienta de compilación de recursos administrados
   ///          asociada con todos los archivos .resx de los que depende esta clase. De lo contrario,
   ///          los diseñadores no podrán interactuar correctamente con los
   ///          recursos adaptados asociados con este formulario.
   /// </summary>
   public ref class Form1 : public System::Windows::Forms::Form
   {
   public:
      Form1(void)
      {
         InitializeComponent();
         //
         //TODO: agregar código de constructor aquí
         //
      }
   protected:
      /// <summary>
      /// Limpiar los recursos que se estén utilizando.
      /// </summary>
      ~Form1()
      {
         if (components)
         {
            delete components;
         }
      }
   private: System::Data::DataSet^  dataSet1;
   private: ZedGraph::ZedGraphControl^  zedGraphControl3;
   private: ZedGraph::ZedGraphControl^  zedGraphControl2;
   private: System::Windows::Forms::ToolStripMenuItem^  todosLosCanalesToolStripMenuItem1;
   private: System::Windows::Forms::ToolStripMenuItem^  canal3ToolStripMenuItem1;
   private: System::Windows::Forms::ToolStripMenuItem^  aceleraciónToolStripMenuItem;
   private: System::Windows::Forms::ToolStripMenuItem^  voltajeToolStripMenuItem;
   private: System::Windows::Forms::ToolStripMenuItem^  canalesToolStripMenuItem;
   private: System::Windows::Forms::ToolStripMenuItem^  canal1ToolStripMenuItem1;
   private: System::Windows::Forms::ToolStripMenuItem^  canal2ToolStripMenuItem1;
   private: ZedGraph::ZedGraphControl^  zedGraphControl1;
   private: System::Windows::Forms::Label^  label5;
   private: System::Windows::Forms::Timer^  timerGraficar;
   private: System::Windows::Forms::TextBox^  txtInicio;
   private: System::Windows::Forms::TextBox^  txtArchivo;
   private: System::Windows::Forms::Label^  label1;
   private: System::Windows::Forms::Label^  label4;
   private: System::Windows::Forms::GroupBox^  grbEventos;
   private: System::Windows::Forms::TextBox^  txtFecha;
   private: System::Windows::Forms::TextBox^  txtSitio;
   private: System::Windows::Forms::Label^  label3;
   private: System::Windows::Forms::TextBox^  txtTranscurrido;
```

```
   private: System::Windows::Forms::Label^  label2;
   private: System::Windows::Forms::SaveFileDialog^  saveFileDialog1;
   private: System::Windows::Forms::ToolStripMenuItem^  ejeYToolStripMenuItem;
   private: System::Windows::Forms::RadioButton^  rdbCanal3;
   private: System::Windows::Forms::ToolStripMenuItem^  númeroDeMuestrasToolStripMenuItem;
   private: System::Windows::Forms::CheckBox^  chkCanales;
   private: System::Windows::Forms::RadioButton^  rdbSegundos;
   private: System::Windows::Forms::RadioButton^  rdbMinutos;
   private: System::Windows::Forms::GroupBox^  grbTiempo;
   private: System::Windows::Forms::NumericUpDown^  numDuraciónMuestreo;
   private: System::Windows::Forms::Label^  lblSampleRate;
   private: System::Windows::Forms::Label^  lblDuraciónMuestreo;
   private: System::Windows::Forms::NumericUpDown^  numSampleRate;
   private: System::Windows::Forms::ToolStripStatusLabel^  toolLabel1;
   private: System::Windows::Forms::RadioButton^  rdbCanal1;
   private: System::Windows::Forms::GroupBox^  grbCanales;
   private: System::Windows::Forms::RadioButton^  rdbCanal2;
   private: System::Windows::Forms::StatusStrip^  statusStrip1;
   private: System::Windows::Forms::ToolStripMenuItem^  visualizarToolStripMenuItem;
   private: System::Windows::Forms::ToolStripMenuItem^  ejesToolStripMenuItem;
   private: System::Windows::Forms::ToolStripMenuItem^  tiempoToolStripMenuItem;
   private: System::Windows::Forms::ToolStripMenuItem^  informaciónGeneralToolStripMenuItem;
   private: System::Windows::Forms::Label^  label6;
   private: System::Windows::Forms::Button^  btnParar;
   private: System::Windows::Forms::Button^  btnIniciar;
   private: System::Windows::Forms::ToolStripMenuItem^  adquirirToolStripMenuItem;
   private: System::Windows::Forms::MenuStrip^  menuStrip1;
   private: System::Windows::Forms::ToolStripMenuItem^  archivoToolStripMenuItem;
   private: System::Windows::Forms::ToolStripMenuItem^  salirToolStripMenuItem;
   private: System::ComponentModel::IContainer^  components;
   private: System::Windows::Forms::ToolStripStatusLabel^  toolStripStatusXY;
   private: System::Windows::Forms::ToolStripStatusLabel^  toolStripStatusScroll;
   private: System::Windows::Forms::CheckBox^  chkGuardar;
   private: System::Windows::Forms::HelpProvider^  helpProvider1;
   private: System::Windows::Forms::Button^  btnAyuda;
   private: System::Windows::Forms::ImageList^  imageList1;
   private: System::Windows::Forms::OpenFileDialog^  openFile;
   private: System::Windows::Forms::Label^  lblGraf;
   private: System::Windows::Forms::ToolTip^  toolTip1;
   private:
      /// <summary>
      /// Variable del diseñador requerida.
      /// </summary>
      PanelBienvenida^ Intro;
      PanelInfoEvento^ Info;
      String^ varOperador;
      String^ varSitio;
      String^ varProyecto;
      bool hError;
      bool continuar;
      Adquisición * adq;            //Crea un objeto de la clase adquisición
      int intTickStart;          //Guarda el número de milisegundos transcurridos desde que se inició el sistema.
      int intCCanal;             //Canal a adquirir
       float64 flTasaMuestreo;         //Tasa de muestreo
       int  intDuraciónMuestreo;    //Tiempo de duración de la toma de datos
      array<double>^ datosCanal1;      //Array donde se lee continuamente el data - canal 1
      array<double>^ datosCanal2;      //Array donde se lee continuamente el data - canal 2
      array<double>^ datosCanal3;      //Array donde se lee continuamente el data - canal 3
      array<double>^ time1;         //Array del tiempo canal 1
      array<double>^ time2;         //Array del tiempo canal 2
      array<double>^ time3;         //Array del tiempo canal 3
      int intIncremento1, intIncremento2, intIncremento3;
      int intCounterCanal1, intCounterCanal2, intCounterCanal3;   //Contadores para graficar;
      int stopTime;              //Duración del muestreo
      int valorMax;              //Valor máximo del eje X
      FileInfo^ fileDatos;       //Objeto FileInfo para crear archivo
      StreamWriter^ strDatos;       //Objeto StreamWriter
      bool bEjexTiempo;          //Variable que controla eje x
          bool bEjeyAceleración;       //Variable que controla eje y

#pragma region Windows Form Designer generated code
      /// <summary>
      /// Método necesario para admitir el Diseñador. No se puede modificar
      /// el contenido del método con el editor de código.
      /// </summary>
      void InitializeComponent(void)
      {
         this->components = (gcnew System::ComponentModel::Container());
         System::ComponentModel::ComponentResourceManager^  resources = (gcnew System::ComponentModel::ComponentResourceManager(Form1::typeid));
         this->dataSet1 = (gcnew System::Data::DataSet());
         this->zedGraphControl3 = (gcnew ZedGraph::ZedGraphControl());
         this->zedGraphControl2 = (gcnew ZedGraph::ZedGraphControl());
         this->todosLosCanalesToolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripMenuItem());
         this->canal3ToolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripMenuItem());
         this->aceleraciónToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
         this->voltajeToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
         this->canalesToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
         this->canal1ToolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripMenuItem());
```

```
this->canal2ToolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->zedGraphControl1 = (gcnew ZedGraph::ZedGraphControl());
this->label5 = (gcnew System::Windows::Forms::Label());
this->timerGraficar = (gcnew System::Windows::Forms::Timer(this->components));
this->txtInicio = (gcnew System::Windows::Forms::TextBox());
this->txtArchivo = (gcnew System::Windows::Forms::TextBox());
this->label1 = (gcnew System::Windows::Forms::Label());
this->label4 = (gcnew System::Windows::Forms::Label());
this->grbEventos = (gcnew System::Windows::Forms::GroupBox());
this->txtFecha = (gcnew System::Windows::Forms::TextBox());
this->txtSitio = (gcnew System::Windows::Forms::TextBox());
this->label3 = (gcnew System::Windows::Forms::Label());
this->txtTranscurrido = (gcnew System::Windows::Forms::TextBox());
this->label2 = (gcnew System::Windows::Forms::Label());
this->saveFileDialog1 = (gcnew System::Windows::Forms::SaveFileDialog());
this->ejeYToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->rdbCanal3 = (gcnew System::Windows::Forms::RadioButton());
this->númeroDeMuestrasToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->chkCanales = (gcnew System::Windows::Forms::CheckBox());
this->rdbSegundos = (gcnew System::Windows::Forms::RadioButton());
this->rdbMinutos = (gcnew System::Windows::Forms::RadioButton());
this->grbTiempo = (gcnew System::Windows::Forms::GroupBox());
this->numDuraciónMuestreo = (gcnew System::Windows::Forms::NumericUpDown());
this->lblSampleRate = (gcnew System::Windows::Forms::Label());
this->lblDuraciónMuestreo = (gcnew System::Windows::Forms::Label());
this->numSampleRate = (gcnew System::Windows::Forms::NumericUpDown());
this->toolLabel1 = (gcnew System::Windows::Forms::ToolStripStatusLabel());
this->rdbCanal1 = (gcnew System::Windows::Forms::RadioButton());
this->grbCanales = (gcnew System::Windows::Forms::GroupBox());
this->rdbCanal2 = (gcnew System::Windows::Forms::RadioButton());
this->statusStrip1 = (gcnew System::Windows::Forms::StatusStrip());
this->toolStripStatusXY = (gcnew System::Windows::Forms::ToolStripStatusLabel());
this->toolStripStatusScroll = (gcnew System::Windows::Forms::ToolStripStatusLabel());
this->visualizarToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->ejesToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->tiempoToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->informaciónGeneralToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->label6 = (gcnew System::Windows::Forms::Label());
this->btnParar = (gcnew System::Windows::Forms::Button());
this->btnIniciar = (gcnew System::Windows::Forms::Button());
this->adquirirToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
this->archivoToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->salirToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->chkGuardar = (gcnew System::Windows::Forms::CheckBox());
this->helpProvider1 = (gcnew System::Windows::Forms::HelpProvider());
this->btnAyuda = (gcnew System::Windows::Forms::Button());
this->imageList1 = (gcnew System::Windows::Forms::ImageList(this->components));
this->openFile = (gcnew System::Windows::Forms::OpenFileDialog());
this->lblGraf = (gcnew System::Windows::Forms::Label());
this->toolTip1 = (gcnew System::Windows::Forms::ToolTip(this->components));
(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->dataSet1))->BeginInit();
this->grbEventos->SuspendLayout();
this->grbTiempo->SuspendLayout();
(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->numDuraciónMuestreo))->BeginInit();
(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->numSampleRate))->BeginInit();
this->grbCanales->SuspendLayout();
this->statusStrip1->SuspendLayout();
this->menuStrip1->SuspendLayout();
this->SuspendLayout();
//
// dataSet1
//
this->dataSet1->DataSetName = L"NewDataSet";
//
// zedGraphControl3
//
this->zedGraphControl3->Location = System::Drawing::Point(357, 493);
this->zedGraphControl3->Margin = System::Windows::Forms::Padding(5, 4, 5, 4);
this->zedGraphControl3->Name = L"zedGraphControl3";
this->zedGraphControl3->ScrollGrace = 0;
this->zedGraphControl3->ScrollMaxX = 0;
this->zedGraphControl3->ScrollMaxY = 0;
this->zedGraphControl3->ScrollMaxY2 = 0;
this->zedGraphControl3->ScrollMinX = 0;
this->zedGraphControl3->ScrollMinY = 0;
this->zedGraphControl3->ScrollMinY2 = 0;
this->zedGraphControl3->Size = System::Drawing::Size(990, 230);
this->zedGraphControl3->TabIndex = 47;
//
// zedGraphControl2
//
this->zedGraphControl2->Location = System::Drawing::Point(357, 263);
this->zedGraphControl2->Margin = System::Windows::Forms::Padding(5, 4, 5, 4);
this->zedGraphControl2->Name = L"zedGraphControl2";
this->zedGraphControl2->ScrollGrace = 0;
this->zedGraphControl2->ScrollMaxX = 0;
this->zedGraphControl2->ScrollMaxY = 0;
```

```
this->zedGraphControl2->ScrollMaxY2 = 0;
this->zedGraphControl2->ScrollMinX = 0;
this->zedGraphControl2->ScrollMinY = 0;
this->zedGraphControl2->ScrollMinY2 = 0;
this->zedGraphControl2->Size = System::Drawing::Size(990, 230);
this->zedGraphControl2->TabIndex = 46;
//
// todosLosCanalesToolStripMenuItem1
//
this->todosLosCanalesToolStripMenuItem1->Name = L"todosLosCanalesToolStripMenuItem1";
this->todosLosCanalesToolStripMenuItem1->Size = System::Drawing::Size(197, 24);
this->todosLosCanalesToolStripMenuItem1->Text = L"Todos los Canales";
this->todosLosCanalesToolStripMenuItem1->Click += gcnew System::EventHandler(this,
   &Form1::todosLosCanalesToolStripMenuItem1_Click);
//
// canal3ToolStripMenuItem1
//
this->canal3ToolStripMenuItem1->Name = L"canal3ToolStripMenuItem1";
this->canal3ToolStripMenuItem1->Size = System::Drawing::Size(197, 24);
this->canal3ToolStripMenuItem1->Text = L"Canal 3";
this->canal3ToolStripMenuItem1->Click += gcnew System::EventHandler(this, &Form1::canal3ToolStripMenuItem1_Click);
//
// aceleraciónToolStripMenuItem
//
this->aceleraciónToolStripMenuItem->Name = L"aceleraciónToolStripMenuItem";
this->aceleraciónToolStripMenuItem->Size = System::Drawing::Size(156, 24);
this->aceleraciónToolStripMenuItem->Text = L"Aceleración";
this->aceleraciónToolStripMenuItem->Click += gcnew System::EventHandler(this, &Form1::aceleraciónToolStripMenuItem_Click);
//
// voltajeToolStripMenuItem
//
this->voltajeToolStripMenuItem->Name = L"voltajeToolStripMenuItem";
this->voltajeToolStripMenuItem->Size = System::Drawing::Size(156, 24);
this->voltajeToolStripMenuItem->Text = L"Voltaje";
this->voltajeToolStripMenuItem->Click += gcnew System::EventHandler(this, &Form1::voltajeToolStripMenuItem_Click);
//
// canalesToolStripMenuItem
//
this->canalesToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(4)
{this->canal1ToolStripMenuItem1,
   this->canal2ToolStripMenuItem1, this->canal3ToolStripMenuItem1, this->todosLosCanalesToolStripMenuItem1});
this->canalesToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^  >(resources->GetObject
   (L"canalesToolStripMenuItem.Image")));
this->canalesToolStripMenuItem->Name = L"canalesToolStripMenuItem";
this->canalesToolStripMenuItem->Size = System::Drawing::Size(172, 24);
this->canalesToolStripMenuItem->Text = L"Canales";
this->canalesToolStripMenuItem->ToolTipText = L"Elija el canal o los canales que desea visualizar";
//
// canal1ToolStripMenuItem1
//
this->canal1ToolStripMenuItem1->Name = L"canal1ToolStripMenuItem1";
this->canal1ToolStripMenuItem1->Size = System::Drawing::Size(197, 24);
this->canal1ToolStripMenuItem1->Text = L"Canal 1";
this->canal1ToolStripMenuItem1->Click += gcnew System::EventHandler(this, &Form1::canal1ToolStripMenuItem1_Click);
//
// canal2ToolStripMenuItem1
//
this->canal2ToolStripMenuItem1->Name = L"canal2ToolStripMenuItem1";
this->canal2ToolStripMenuItem1->Size = System::Drawing::Size(197, 24);
this->canal2ToolStripMenuItem1->Text = L"Canal 2";
this->canal2ToolStripMenuItem1->Click += gcnew System::EventHandler(this, &Form1::canal2ToolStripMenuItem1_Click);
//
// zedGraphControl1
//
this->zedGraphControl1->Location = System::Drawing::Point(357, 33);
this->zedGraphControl1->Margin = System::Windows::Forms::Padding(5, 4, 5, 4);
this->zedGraphControl1->Name = L"zedGraphControl1";
this->zedGraphControl1->ScrollGrace = 0;
this->zedGraphControl1->ScrollMaxX = 0;
this->zedGraphControl1->ScrollMaxY = 0;
this->zedGraphControl1->ScrollMaxY2 = 0;
this->zedGraphControl1->ScrollMinX = 0;
this->zedGraphControl1->ScrollMinY = 0;
this->zedGraphControl1->ScrollMinY2 = 0;
this->zedGraphControl1->Size = System::Drawing::Size(990, 230);
this->zedGraphControl1->TabIndex = 45;
this->zedGraphControl1->MouseMoveEvent += gcnew ZedGraph::ZedGraphControl::ZedMouseEventHandler(this,
   &Form1::zedGraphControl1_MouseMoveEvent);
this->zedGraphControl1->ScrollProgressEvent += gcnew ZedGraph::ZedGraphControl::ScrollProgressHandler(this,
   &Form1::zedGraphControl1_ScrollProgressEvent);
//
// label5
//
this->label5->AutoSize = true;
this->label5->Location = System::Drawing::Point(18, 134);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(99, 17);
this->label5->TabIndex = 45;
```

```
this->label5->Text = L"Hora de inicio:";
//
// timerGraficar
//
this->timerGraficar->Tick += gcnew System::EventHandler(this, &Form1::timerGraficar_Tick);
//
// txtInicio
//
this->txtInicio->Location = System::Drawing::Point(168, 129);
this->txtInicio->Name = L"txtInicio";
this->txtInicio->ReadOnly = true;
this->txtInicio->Size = System::Drawing::Size(109, 22);
this->txtInicio->TabIndex = 46;
this->toolTip1->SetToolTip(this->txtInicio, L"Hora de inicio del evento de adquisición");
//
// txtArchivo
//
this->txtArchivo->Location = System::Drawing::Point(83, 101);
this->txtArchivo->Name = L"txtArchivo";
this->txtArchivo->ReadOnly = true;
this->txtArchivo->Size = System::Drawing::Size(194, 22);
this->txtArchivo->TabIndex = 44;
this->toolTip1->SetToolTip(this->txtArchivo, L"Nombre del archivo donde se guardarán los datos adquiridos.\r\nPuede "
   L"cambiarlo en el cuadro de diálogo Guardar como, que aparece al dar click en el botón Iniciar.");
//
// label1
//
this->label1->AutoSize = true;
this->label1->Location = System::Drawing::Point(18, 101);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(59, 17);
this->label1->TabIndex = 43;
this->label1->Text = L"Archivo:";
//
// label4
//
this->label4->AutoSize = true;
this->label4->Location = System::Drawing::Point(18, 68);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(51, 17);
this->label4->TabIndex = 41;
this->label4->Text = L"Fecha:";
//
// grbEventos
//
this->grbEventos->Controls->Add(this->txtInicio);
this->grbEventos->Controls->Add(this->label5);
this->grbEventos->Controls->Add(this->txtArchivo);
this->grbEventos->Controls->Add(this->label1);
this->grbEventos->Controls->Add(this->txtFecha);
this->grbEventos->Controls->Add(this->label4);
this->grbEventos->Controls->Add(this->txtSitio);
this->grbEventos->Controls->Add(this->label3);
this->grbEventos->Controls->Add(this->txtTranscurrido);
this->grbEventos->Controls->Add(this->label2);
this->grbEventos->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->grbEventos->Location = System::Drawing::Point(19, 412);
this->grbEventos->Name = L"grbEventos";
this->grbEventos->Size = System::Drawing::Size(295, 202);
this->grbEventos->TabIndex = 48;
this->grbEventos->TabStop = false;
this->grbEventos->Text = L"Evento";
this->toolTip1->SetToolTip(this->grbEventos, L"Para editar información del evento ir a menú Adquirir opción Información General");
//
// txtFecha
//
this->txtFecha->Location = System::Drawing::Point(83, 65);
this->txtFecha->Name = L"txtFecha";
this->txtFecha->ReadOnly = true;
this->txtFecha->Size = System::Drawing::Size(194, 22);
this->txtFecha->TabIndex = 42;
this->toolTip1->SetToolTip(this->txtFecha, L"Fecha actual");
//
// txtSitio
//
this->txtSitio->Location = System::Drawing::Point(83, 30);
this->txtSitio->Name = L"txtSitio";
this->txtSitio->ReadOnly = true;
this->txtSitio->Size = System::Drawing::Size(194, 22);
this->txtSitio->TabIndex = 40;
this->toolTip1->SetToolTip(this->txtSitio, L"Para editar ir a menú Adquirir, opción Información General");
//
// label3
//
this->label3->AutoSize = true;
this->label3->Location = System::Drawing::Point(18, 35);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(39, 17);
```

```
this->label3->TabIndex = 39;
this->label3->Text = L"Sitio:";
//
// txtTranscurrido
//
this->txtTranscurrido->Location = System::Drawing::Point(202, 164);
this->txtTranscurrido->Name = L"txtTranscurrido";
this->txtTranscurrido->ReadOnly = true;
this->txtTranscurrido->Size = System::Drawing::Size(75, 22);
this->txtTranscurrido->TabIndex = 38;
this->toolTip1->SetToolTip(this->txtTranscurrido, L"Tiempo en segundos desde que inicia la adquisición");
//
// label2
//
this->label2->AutoSize = true;
this->label2->Location = System::Drawing::Point(18, 167);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(160, 17);
this->label2->TabIndex = 5;
this->label2->Text = L"Tiempo transcurrido (s):";
//
// saveFileDialog1
//
this->saveFileDialog1->DefaultExt = L"txt";
this->saveFileDialog1->Filter = L"Archivo de texto (*.txt)|*.txt";
this->saveFileDialog1->OverwritePrompt = false;
this->saveFileDialog1->Title = L"Guardar Archivo Como";
//
// ejeYToolStripMenuItem
//
this->ejeYToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(2)
{this->aceleraciónToolStripMenuItem,
   this->voltajeToolStripMenuItem});
this->ejeYToolStripMenuItem->Name = L"ejeYToolStripMenuItem";
this->ejeYToolStripMenuItem->Size = System::Drawing::Size(172, 24);
this->ejeYToolStripMenuItem->Text = L"Eje ordenadas";
this->ejeYToolStripMenuItem->ToolTipText = L"Seleccione unidades de aceleración o voltaje para \r\nel "
   L"eje de las ordenadas (eje y) de las gráficas.";
//
// rdbCanal3
//
this->rdbCanal3->AutoSize = true;
this->rdbCanal3->Location = System::Drawing::Point(21, 82);
this->rdbCanal3->Name = L"rdbCanal3";
this->rdbCanal3->Size = System::Drawing::Size(121, 21);
this->rdbCanal3->TabIndex = 2;
this->rdbCanal3->TabStop = true;
this->rdbCanal3->Text = L"Canal 3 - Eje z";
this->rdbCanal3->UseVisualStyleBackColor = true;
this->rdbCanal3->CheckedChanged += gcnew System::EventHandler(this, &Form1::rdbCanal3_CheckedChanged);
//
// númeroDeMuestrasToolStripMenuItem
//
this->númeroDeMuestrasToolStripMenuItem->Name = L"númeroDeMuestrasToolStripMenuItem";
this->númeroDeMuestrasToolStripMenuItem->Size = System::Drawing::Size(216, 24);
this->númeroDeMuestrasToolStripMenuItem->Text = L"Número de muestras";
this->númeroDeMuestrasToolStripMenuItem->Click += gcnew System::EventHandler(this,
   &Form1::númeroDeMuestrasToolStripMenuItem_Click);
//
// chkCanales
//
this->chkCanales->AutoSize = true;
this->chkCanales->Checked = true;
this->chkCanales->CheckState = System::Windows::Forms::CheckState::Checked;
this->chkCanales->Location = System::Drawing::Point(21, 119);
this->chkCanales->Name = L"chkCanales";
this->chkCanales->Size = System::Drawing::Size(145, 21);
this->chkCanales->TabIndex = 3;
this->chkCanales->Text = L"Todos los canales";
this->chkCanales->UseVisualStyleBackColor = true;
this->chkCanales->CheckedChanged += gcnew System::EventHandler(this, &Form1::chkCanales_CheckedChanged);
//
// rdbSegundos
//
this->rdbSegundos->AutoSize = true;
this->rdbSegundos->Checked = true;
this->rdbSegundos->Location = System::Drawing::Point(97, 141);
this->rdbSegundos->Name = L"rdbSegundos";
this->rdbSegundos->Size = System::Drawing::Size(36, 21);
this->rdbSegundos->TabIndex = 43;
this->rdbSegundos->TabStop = true;
this->rdbSegundos->Text = L"s";
this->toolTip1->SetToolTip(this->rdbSegundos, L"Segundos");
this->rdbSegundos->UseVisualStyleBackColor = true;
this->rdbSegundos->CheckedChanged += gcnew System::EventHandler(this, &Form1::rdbSegundos_CheckedChanged);
//
// rdbMinutos
//
```

```
this->rdbMinutos->AutoSize = true;
this->rdbMinutos->Location = System::Drawing::Point(21, 141);
this->rdbMinutos->Name = L"rdbMinutos";
this->rdbMinutos->Size = System::Drawing::Size(51, 21);
this->rdbMinutos->TabIndex = 42;
this->rdbMinutos->Text = L"min";
this->toolTip1->SetToolTip(this->rdbMinutos, L"Minutos");
this->rdbMinutos->UseVisualStyleBackColor = true;
this->rdbMinutos->CheckedChanged += gcnew System::EventHandler(this, &Form1::rdbMinutos_CheckedChanged);
//
// grbTiempo
//
this->grbTiempo->BackColor = System::Drawing::SystemColors::Control;
this->grbTiempo->Controls->Add(this->rdbSegundos);
this->grbTiempo->Controls->Add(this->rdbMinutos);
this->grbTiempo->Controls->Add(this->numDuraciónMuestreo);
this->grbTiempo->Controls->Add(this->lblSampleRate);
this->grbTiempo->Controls->Add(this->lblDuraciónMuestreo);
this->grbTiempo->Controls->Add(this->numSampleRate);
this->grbTiempo->Location = System::Drawing::Point(19, 218);
this->grbTiempo->Name = L"grbTiempo";
this->grbTiempo->Size = System::Drawing::Size(190, 174);
this->grbTiempo->TabIndex = 41;
this->grbTiempo->TabStop = false;
this->grbTiempo->Text = L"Parámetros de Tiempo";
//
// numDuraciónMuestreo
//
this->numDuraciónMuestreo->Location = System::Drawing::Point(21, 113);
this->numDuraciónMuestreo->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) {60, 0, 0, 0});
this->numDuraciónMuestreo->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) {3, 0, 0, 0});
this->numDuraciónMuestreo->Name = L"numDuraciónMuestreo";
this->numDuraciónMuestreo->Size = System::Drawing::Size(89, 22);
this->numDuraciónMuestreo->TabIndex = 37;
this->numDuraciónMuestreo->TextAlign = System::Windows::Forms::HorizontalAlignment::Right;
this->toolTip1->SetToolTip(this->numDuraciónMuestreo, L"Escoja el lapso de tiempo que durará la adquisición");
this->numDuraciónMuestreo->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) {10, 0, 0, 0});
//
// lblSampleRate
//
this->lblSampleRate->AutoSize = true;
this->lblSampleRate->Location = System::Drawing::Point(15, 28);
this->lblSampleRate->Name = L"lblSampleRate";
this->lblSampleRate->Size = System::Drawing::Size(154, 17);
this->lblSampleRate->TabIndex = 3;
this->lblSampleRate->Text = L"Tasa de muestreo (Hz)";
//
// lblDuraciónMuestreo
//
this->lblDuraciónMuestreo->AutoSize = true;
this->lblDuraciónMuestreo->Location = System::Drawing::Point(15, 94);
this->lblDuraciónMuestreo->Name = L"lblDuraciónMuestreo";
this->lblDuraciónMuestreo->Size = System::Drawing::Size(134, 17);
this->lblDuraciónMuestreo->TabIndex = 36;
this->lblDuraciónMuestreo->Text = L"Lapso de tiempo (s)";
//
// numSampleRate
//
this->numSampleRate->Increment = System::Decimal(gcnew cli::array< System::Int32 >(4) {10, 0, 0, 0});
this->numSampleRate->Location = System::Drawing::Point(21, 49);
this->numSampleRate->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) {10000, 0, 0, 0});
this->numSampleRate->Name = L"numSampleRate";
this->numSampleRate->Size = System::Drawing::Size(89, 22);
this->numSampleRate->TabIndex = 1;
this->numSampleRate->TextAlign = System::Windows::Forms::HorizontalAlignment::Right;
this->toolTip1->SetToolTip(this->numSampleRate, L"Escoja frecuencia de muestreo para realizar la adquisición de datos");
this->numSampleRate->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) {200, 0, 0, 0});
//
// toolLabel1
//
this->toolLabel1->Name = L"toolLabel1";
this->toolLabel1->Size = System::Drawing::Size(120, 23);
this->toolLabel1->Text = L"Listo para adquirir";
//
// rdbCanal1
//
this->rdbCanal1->AutoSize = true;
this->rdbCanal1->Location = System::Drawing::Point(21, 30);
this->rdbCanal1->Name = L"rdbCanal1";
this->rdbCanal1->Size = System::Drawing::Size(120, 21);
this->rdbCanal1->TabIndex = 0;
this->rdbCanal1->TabStop = true;
this->rdbCanal1->Text = L"Canal 1 - Eje x";
this->rdbCanal1->UseVisualStyleBackColor = true;
this->rdbCanal1->CheckedChanged += gcnew System::EventHandler(this, &Form1::rdbCanal1_CheckedChanged);
//
// grbCanales
//
```

```
this->grbCanales->Controls->Add(this->chkCanales);
this->grbCanales->Controls->Add(this->rdbCanal3);
this->grbCanales->Controls->Add(this->rdbCanal2);
this->grbCanales->Controls->Add(this->rdbCanal1);
this->grbCanales->Location = System::Drawing::Point(19, 48);
this->grbCanales->Name = L"grbCanales";
this->grbCanales->Size = System::Drawing::Size(190, 155);
this->grbCanales->TabIndex = 40;
this->grbCanales->TabStop = false;
this->grbCanales->Text = L"Canales de entrada";
this->toolTip1->SetToolTip(this->grbCanales, L"Seleccione el canal/canales a adquirir");
//
// rdbCanal2
//
this->rdbCanal2->AutoSize = true;
this->rdbCanal2->Location = System::Drawing::Point(21, 55);
this->rdbCanal2->Name = L"rdbCanal2";
this->rdbCanal2->Size = System::Drawing::Size(121, 21);
this->rdbCanal2->TabIndex = 1;
this->rdbCanal2->TabStop = true;
this->rdbCanal2->Text = L"Canal 2 - Eje y";
this->rdbCanal2->UseVisualStyleBackColor = true;
this->rdbCanal2->CheckedChanged += gcnew System::EventHandler(this, &Form1::rdbCanal2_CheckedChanged);
//
// statusStrip1
//
this->statusStrip1->Font = (gcnew System::Drawing::Font(L"Segoe UI", 7.8F));
this->statusStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(3)
{this->toolLabel1, this->toolStripStatusXY,
   this->toolStripStatusScroll});
this->statusStrip1->Location = System::Drawing::Point(0, 748);
this->statusStrip1->Name = L"statusStrip1";
this->statusStrip1->Size = System::Drawing::Size(1362, 28);
this->statusStrip1->TabIndex = 39;
this->statusStrip1->Text = L"statusStrip1";
//
// toolStripStatusXY
//
this->toolStripStatusXY->BorderSides = static_cast<System::Windows::Forms::ToolStripStatusLabelBorderSides>
   ((((System::Windows::Forms::ToolStripStatusLabelBorderSides::Left | System::Windows::Forms::
   ToolStripStatusLabelBorderSides::Top)
   | System::Windows::Forms::ToolStripStatusLabelBorderSides::Right)
   | System::Windows::Forms::ToolStripStatusLabelBorderSides::Bottom));
this->toolStripStatusXY->BorderStyle = System::Windows::Forms::Border3DStyle::SunkenOuter;
this->toolStripStatusXY->Name = L"toolStripStatusXY";
this->toolStripStatusXY->RightToLeft = System::Windows::Forms::RightToLeft::No;
this->toolStripStatusXY->Size = System::Drawing::Size(143, 23);
this->toolStripStatusXY->Text = L"toolStripStatusLabel1";
//
// toolStripStatusScroll
//
this->toolStripStatusScroll->Name = L"toolStripStatusScroll";
this->toolStripStatusScroll->Size = System::Drawing::Size(139, 23);
this->toolStripStatusScroll->Text = L"toolStripStatusLabel1";
//
// visualizarToolStripMenuItem
//
this->visualizarToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(3)
{this->ejesToolStripMenuItem,
   this->ejeYToolStripMenuItem, this->canalesToolStripMenuItem});
this->visualizarToolStripMenuItem->Name = L"visualizarToolStripMenuItem";
this->visualizarToolStripMenuItem->ShortcutKeys = static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Alt
   | System::Windows::Forms::Keys::V));
this->visualizarToolStripMenuItem->Size = System::Drawing::Size(84, 24);
this->visualizarToolStripMenuItem->Text = L"&Visualizar";
this->visualizarToolStripMenuItem->TextImageRelation = System::Windows::Forms::TextImageRelation::Overlay;
this->visualizarToolStripMenuItem->ToolTipText = L"(Alt+V)";
//
// ejesToolStripMenuItem
//
this->ejesToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(2)
{this->tiempoToolStripMenuItem,
   this->números DeMuestrasToolStripMenuItem});
this->ejesToolStripMenuItem->Name = L"ejesToolStripMenuItem";
this->ejesToolStripMenuItem->Size = System::Drawing::Size(172, 24);
this->ejesToolStripMenuItem->Text = L"Eje abcisas";
this->ejesToolStripMenuItem->ToolTipText = L"Escoja si desea mostrar tiempo o número de muestras \r\nen el eje de "
   L"las abcisas (eje x) de las gráficas.";
//
// tiempoToolStripMenuItem
//
this->tiempoToolStripMenuItem->Name = L"tiempoToolStripMenuItem";
this->tiempoToolStripMenuItem->Size = System::Drawing::Size(216, 24);
this->tiempoToolStripMenuItem->Text = L"Tiempo";
this->tiempoToolStripMenuItem->Click += gcnew System::EventHandler(this, &Form1::tiempoToolStripMenuItem_Click);
//
// informaciónGeneralToolStripMenuItem
//
```

```
this->informaciónGeneralToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^  >
   (resources->GetObject(L"informaciónGeneralToolStripMenuItem.Image")));
this->informaciónGeneralToolStripMenuItem->Name = L"informaciónGeneralToolStripMenuItem";
this->informaciónGeneralToolStripMenuItem->Size = System::Drawing::Size(222, 24);
this->informaciónGeneralToolStripMenuItem->Text = L"Información General...";
this->informaciónGeneralToolStripMenuItem->ToolTipText = L"Seleccione para editar la información del evento\r\na registrar.";
this->informaciónGeneralToolStripMenuItem->Click += gcnew System::EventHandler(this, &Form1::informaciónGeneralToolStripMenuItem_Click);
//
// label6
//
this->label6->AutoSize = true;
this->label6->Location = System::Drawing::Point(261, 582);
this->label6->Name = L"label6";
this->label6->Size = System::Drawing::Size(15, 17);
this->label6->TabIndex = 49;
this->label6->Text = L"s";
//
// btnParar
//
this->btnParar->BackColor = System::Drawing::Color::IndianRed;
this->btnParar->Location = System::Drawing::Point(187, 676);
this->btnParar->Name = L"btnParar";
this->btnParar->Size = System::Drawing::Size(74, 32);
this->btnParar->TabIndex = 43;
this->btnParar->Text = L"De&tener";
this->toolTip1->SetToolTip(this->btnParar, L"De click para detener adquisición");
this->btnParar->UseVisualStyleBackColor = false;
this->btnParar->Click += gcnew System::EventHandler(this, &Form1::btnParar_Click);
//
// btnIniciar
//
this->btnIniciar->BackColor = System::Drawing::Color::LimeGreen;
this->btnIniciar->Location = System::Drawing::Point(62, 676);
this->btnIniciar->Name = L"btnIniciar";
this->btnIniciar->Size = System::Drawing::Size(74, 32);
this->btnIniciar->TabIndex = 42;
this->btnIniciar->Text = L"&Iniciar";
this->toolTip1->SetToolTip(this->btnIniciar, L"De click para iniciar adquisición");
this->btnIniciar->UseVisualStyleBackColor = false;
this->btnIniciar->Click += gcnew System::EventHandler(this, &Form1::btnIniciar_Click);
//
// adquirirToolStripMenuItem
//
this->adquirirToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(1)
{this->informaciónGeneralToolStripMenuItem});
this->adquirirToolStripMenuItem->Name = L"adquirirToolStripMenuItem";
this->adquirirToolStripMenuItem->ShortcutKeys = static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Alt
   | System::Windows::Forms::Keys::D));
this->adquirirToolStripMenuItem->Size = System::Drawing::Size(75, 24);
this->adquirirToolStripMenuItem->Text = L"A&dquirir";
this->adquirirToolStripMenuItem->ToolTipText = L"Alt+D";
//
// menuStrip1
//
this->menuStrip1->BackColor = System::Drawing::SystemColors::Control;
this->menuStrip1->Font = (gcnew System::Drawing::Font(L"Segoe UI", 9));
this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(3)
{this->archivoToolStripMenuItem,
   this->adquirirToolStripMenuItem, this->visualizarToolStripMenuItem});
this->menuStrip1->Location = System::Drawing::Point(0, 0);
this->menuStrip1->Name = L"menuStrip1";
this->menuStrip1->Size = System::Drawing::Size(1362, 28);
this->menuStrip1->TabIndex = 44;
this->menuStrip1->Text = L"menuStrip1";
//
// archivoToolStripMenuItem
//
this->archivoToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(1)
{this->salirToolStripMenuItem});
this->archivoToolStripMenuItem->Name = L"archivoToolStripMenuItem";
this->archivoToolStripMenuItem->Size = System::Drawing::Size(71, 24);
this->archivoToolStripMenuItem->Text = L"&Archivo";
this->archivoToolStripMenuItem->ToolTipText = L"Alt+A";
//
// salirToolStripMenuItem
//
this->salirToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^  >
   (resources->GetObject(L"salirToolStripMenuItem.Image")));
this->salirToolStripMenuItem->Name = L"salirToolStripMenuItem";
this->salirToolStripMenuItem->Size = System::Drawing::Size(107, 24);
this->salirToolStripMenuItem->Text = L"Salir";
this->salirToolStripMenuItem->Click += gcnew System::EventHandler(this, &Form1::salirToolStripMenuItem_Click);
//
// chkGuardar
//
this->chkGuardar->AutoSize = true;
this->chkGuardar->Location = System::Drawing::Point(19, 631);
this->chkGuardar->Name = L"chkGuardar";
```

```
this->chkGuardar->Size = System::Drawing::Size(192, 21);
this->chkGuardar->TabIndex = 50;
this->chkGuardar->Text = L"Guardar datos en archivo";
this->toolTip1->SetToolTip(this->chkGuardar, L"Seleccione si desea guardar los datos \r\na adquirir en un archivo de texto.\r\n");
this->chkGuardar->UseVisualStyleBackColor = true;
//
// helpProvider1
//
this->helpProvider1->HelpNamespace = L"C:\\Users\\Maria Jose\\Documents\\Visual Studio 2008\\Projects\\InterfazAdquisición\\Rel"
   L"ease\\Ayuda.chm";
//
// btnAyuda
//
this->btnAyuda->BackColor = System::Drawing::SystemColors::Control;
this->btnAyuda->Cursor = System::Windows::Forms::Cursors::Help;
this->btnAyuda->FlatAppearance->BorderColor = System::Drawing::SystemColors::Control;
this->btnAyuda->FlatStyle = System::Windows::Forms::FlatStyle::Flat;
this->btnAyuda->Font = (gcnew System::Drawing::Font(L"Segoe UI", 9, System::Drawing::FontStyle::Regular,
   System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(0)));
this->helpProvider1->SetHelpNavigator(this->btnAyuda, System::Windows::Forms::HelpNavigator::TableOfContents);
this->btnAyuda->ImageList = this->imageList1;
this->btnAyuda->Location = System::Drawing::Point(234, 0);
this->btnAyuda->Name = L"btnAyuda";
this->helpProvider1->SetShowHelp(this->btnAyuda, true);
this->btnAyuda->Size = System::Drawing::Size(71, 31);
this->btnAyuda->TabIndex = 51;
this->btnAyuda->Text = L"A&yuda";
this->toolTip1->SetToolTip(this->btnAyuda, L"De click para abrir archivo de ayuda o ALT+Y.");
this->btnAyuda->UseVisualStyleBackColor = false;
this->btnAyuda->Click += gcnew System::EventHandler(this, &Form1::btnAyuda_Click);
//
// imageList1
//
this->imageList1->ImageStream = (cli::safe_cast<System::Windows::Forms::ImageListStreamer^
>(resources->GetObject(L"imageList1.ImageStream")));
this->imageList1->TransparentColor = System::Drawing::Color::Transparent;
this->imageList1->Images->SetKeyName(0, L"help.png");
//
// openFile
//
this->openFile->Filter = L"Archivos de texto (*.txt)|*.txt";
this->openFile->Title = L"Abrir registro...";
//
// lblGraf
//
this->lblGraf->AutoSize = true;
this->lblGraf->Location = System::Drawing::Point(736, 8);
this->lblGraf->Name = L"lblGraf";
this->lblGraf->Size = System::Drawing::Size(273, 17);
this->lblGraf->TabIndex = 52;
this->lblGraf->Text = L"Gráficas de visualización canales: 1, 2 y 3";
//
// toolTip1
//
this->toolTip1->IsBalloon = true;
//
// Form1
//
this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(1362, 776);
this->Controls->Add(this->lblGraf);
this->Controls->Add(this->btnAyuda);
this->Controls->Add(this->chkGuardar);
this->Controls->Add(this->zedGraphControl3);
this->Controls->Add(this->zedGraphControl2);
this->Controls->Add(this->zedGraphControl1);
this->Controls->Add(this->grbEventos);
this->Controls->Add(this->grbTiempo);
this->Controls->Add(this->grbCanales);
this->Controls->Add(this->statusStrip1);
this->Controls->Add(this->label6);
this->Controls->Add(this->btnParar);
this->Controls->Add(this->btnIniciar);
this->Controls->Add(this->menuStrip1);
this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::Fixed3D;
this->MaximizeBox = false;
this->Name = L"Form1";
this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;
this->Text = L"Programa Adquisición y Registro";
this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load);
(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->dataSet1))->EndInit();
this->grbEventos->ResumeLayout(false);
this->grbEventos->PerformLayout();
this->grbTiempo->ResumeLayout(false);
this->grbTiempo->PerformLayout();
(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->numDuraciónMuestreo))->EndInit();
(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->numSampleRate))->EndInit();
```

```
         this->grbCanales->ResumeLayout(false);
         this->grbCanales->PerformLayout();
         this->statusStrip1->ResumeLayout(false);
         this->statusStrip1->PerformLayout();
         this->menuStrip1->ResumeLayout(false);
         this->menuStrip1->PerformLayout();
         this->ResumeLayout(false);
         this->PerformLayout();
      }
#pragma endregion

private: System::Void Form1_Load(System::Object^  sender, System::EventArgs^  e)
       {
          //Crea objeto del panel de información
          Info = gcnew PanelInfoEvento;
          //Inicialización variables globales
          intCCanal = 0; intCounterCanal1 = 0; intCounterCanal2 = 0; intCounterCanal3 = 0;
          intIncremento1 = 0; intIncremento2 = 0; intIncremento3 = 0;
          stopTime = 10;
          bEjexTiempo = true; bEjeyAceleración = false; hError = false;
          //Inicialización arrays para datos de canal
          datosCanal1 = gcnew array<double> (10000);
          datosCanal2 = gcnew array<double> (10000);
          datosCanal3 = gcnew array<double> (10000);
          for(int i = 0; i< datosCanal1->Length; i++)
          {
             datosCanal1->SetValue(nullptr,i);
             datosCanal2->SetValue(nullptr,i);
             datosCanal3->SetValue(nullptr,i);
          }
          //Titulos y detalles de las gráficas
          DiseñoGráfica();
          //Crea un objeto de la clase Adquisición
          adq = new Adquisición();
          //Inicialización del array data de la clase Adquisición
          int32 height = size; int32 width = 2;
          for (int n=0;n<height;n++)
          {
             for (int m=0;m<width;m++)
             {
                adq->data[n*width+m] = 0;
             }
          }
          //Configuración timer usado para graficar
          timerGraficar->Interval = 1000;          //Intervalo de 1 segundo para el timer
          intTickStart = Environment::TickCount;
          timerGraficar->Start();
          //Inicialización de los controles de canal
          chkCanales->Checked = true;
          rdbCanal1->Enabled = false;
          rdbCanal2->Enabled = false;
          rdbCanal1->Checked = false;
          rdbCanal3->Enabled = false;
          intCCanal = 4;
          //Inicialización de los controles parámetros de tiempo
          rdbSegundos->Checked = true;
          //Inicialización menú Visualizar
          todosLosCanalesToolStripMenuItem1->Checked = true;
          if(bEjexTiempo == true)         //Eje x tiempo
          {
             tiempoToolStripMenuItem->Checked = true;
             númeroDeMuestrasToolStripMenuItem->Checked = false;
          }
          else                      //Eje x número de muestras
          {
             tiempoToolStripMenuItem->Checked = false;
             númeroDeMuestrasToolStripMenuItem->Checked = true;
          }
          if(bEjeyAceleración == true)    //Eje y aceleración
          {
             voltajeToolStripMenuItem->Checked = false;
             aceleraciónToolStripMenuItem->Checked = true;
          }
          else                      //Eje y voltaje
          {
             voltajeToolStripMenuItem->Checked = true;
             aceleraciónToolStripMenuItem->Checked = false;
          }
          //Inicialización groupBox Evento
          DateTime infoTime = DateTime::Now;
          txtFecha->Text = infoTime.ToString("D");
          //txtHora->Text = infoTime.ToString("T");
          txtArchivo->Text = String::Concat("Datos ",infoTime.ToString("D"));
          grbEventos->Enabled = true;
          varSitio = gcnew String("Universidad de Nariño");
          txtSitio->Text = varSitio;
          varOperador = gcnew String("GIS");
          varProyecto = gcnew String("Microzonificación Sísmica San Juan de Pasto");
```

```
            //Nombre del sitio
            varSitio = Info->txtSitio2->Text;
            //Nombre del operador
            varOperador = Info->txtOperador2->Text;
            //Nombre del proyecto
            varProyecto = Info->txtProyecto2->Text;
            //Inicialización botones
            btnParar->Enabled = false;
            //Crea la carpeta Adquisición dentro de la carpeta personal del usuario,
            //en esta carpeta se guardarán los datos adquiridos
            String^ strMisDocumentos = Environment::GetFolderPath(Environment::SpecialFolder::Personal);
            String^ path = String::Concat(strMisDocumentos, "\\Adquisición y Registro");
            try
            {
               //Determina si el directorio existe
               if(Directory::Exists(path) == false)
               {
                  DirectoryInfo^ di = Directory::CreateDirectory(path);
               }
            }
            catch (Exception^ e)
            {
               MessageBox::Show(String::Concat("Falló al crear la carpeta Adquisición.",e));
            }
            //Inicialización cuadro de diálogo "Guardar como"
            DateTime fechaActual = DateTime::Now;
            //Define el campo nombre de archivo por defecto como Datos y la fecha actual
            saveFileDialog1->FileName = String::Concat("Datos ", fechaActual.ToString("D"));
            //Define el directorio por defecto donde se guardarán  los datos
            saveFileDialog1->InitialDirectory = path;
            //Inicialización barra de estado
            toolStripStatusXY->Text = "";
            toolStripStatusScroll->Text = "";
            //Inicialización control guardar
            chkGuardar->Checked = true;
         }

/*****************************************************/
/***  Creación y configuración de las gráficas    ***/
/***            para los 3 canales                 ***/
/*****************************************************/
private: Void DiseñoGráfica()
         {
            //Configuración Panel1->Canal1
            ConfDesplazar(zedGraphControl1);
            GraphPane^ myPane = zedGraphControl1->GraphPane;
            //Título de la Gráfica
            myPane->Title->Text = "Canal 1 - Eje x";
            myPane->Title->FontSpec->Size = 22.0F;
            //Cambia el color del titulo
            myPane->Title->FontSpec->FontColor = Color::Black;
            //Agrega grid líneas de color gris
            myPane->XAxis->MajorGrid->IsVisible = true;
            myPane->YAxis->MajorGrid->IsVisible = true;
            myPane->XAxis->MajorGrid->Color = Color::Black;
            myPane->YAxis->MajorGrid->Color = Color::Black;
            myPane->XAxis->MinorGrid->IsVisible = true;
            myPane->XAxis->Scale->FontSpec->Size = 20.0F;
            myPane->YAxis->Scale->FontSpec->Size = 20.0F;

            VariableEjeX(zedGraphControl1);
            VariableEjeY(zedGraphControl1);

            //Se genera la curva, guarda 30000 points - "capacity".
            //RollingPointPairList es una clase de almacenamiento
            RollingPointPairList^ list = gcnew RollingPointPairList(30000);
            LineItem^ curve = myPane->AddCurve("",list, Color::Blue, SymbolType::None);
            //Reconfigura los ejes
            zedGraphControl1->AxisChange();

            //Configuración Panel2->Canal2
            ConfDesplazar(zedGraphControl2);
            GraphPane^ myPane2 = zedGraphControl2->GraphPane;
            //Fija los titulos
            myPane2->Title->Text = "Canal 2 - Eje y";
            myPane2->Title->FontSpec->Size = 22.0F;
            //Cambia el color del titulo
            myPane2->Title->FontSpec->FontColor = Color::Black;
            //Agrega grid líneas de color gris
            myPane2->XAxis->MajorGrid->IsVisible = true;
            myPane2->YAxis->MajorGrid->IsVisible = true;
            myPane2->XAxis->MajorGrid->Color = Color::Black;
            myPane2->YAxis->MajorGrid->Color = Color::Black;
            myPane2->XAxis->MinorGrid->IsVisible = true;
            myPane2->XAxis->Scale->FontSpec->Size = 20.0F;
            myPane2->YAxis->Scale->FontSpec->Size = 20.0F;

            VariableEjeX(zedGraphControl2);
```

```
            VariableEjeY(zedGraphControl2);

            RollingPointPairList^ list2 = gcnew RollingPointPairList(30000);
            LineItem^ curve2 = myPane2->AddCurve("",list2, Color::Blue, SymbolType::None);
            zedGraphControl2->AxisChange();

            //Configuración Panel3 Canal3
            ConfDesplazar(zedGraphControl3);
            GraphPane^ myPane3 = zedGraphControl3->GraphPane;
            //Titulo de la gráfica
            myPane3->Title->Text = "Canal 3 - Eje z";
            myPane3->Title->FontSpec->Size = 22.0F;
            //Cambia el color del titulo
            myPane3->Title->FontSpec->FontColor = Color::Black;
            //Agrega grid líneas de color gris
            myPane3->XAxis->MajorGrid->IsVisible = true;
            myPane3->YAxis->MajorGrid->IsVisible = true;
            myPane3->XAxis->MinorGrid->IsVisible = true;
            myPane3->XAxis->MajorGrid->Color = Color::Black;
            myPane3->YAxis->MajorGrid->Color = Color::Black;
            myPane3->XAxis->Scale->FontSpec->Size = 20.0F;
            myPane3->YAxis->Scale->FontSpec->Size = 20.0F;

            VariableEjeX(zedGraphControl3);
            VariableEjeY(zedGraphControl3);

            RollingPointPairList^ list3 = gcnew RollingPointPairList(30000);
            LineItem^ curve3 = myPane3->AddCurve("",list3, Color::Blue, SymbolType::None);
            zedGraphControl3->AxisChange();

            //Guarda el tiempo de inicio como referencia
            intTickStart = Environment::TickCount;
        }
    //Configuración scroll y zoom en las gráficas
    private: Void ConfDesplazar(ZedGraphControl^ zgc)
        {
            //Muestra la barra de desplazamiento horizontal
            zgc->IsShowHScrollBar = true;
            //Fija manualmente el rango de datos de la barra
            zgc->ScrollMinX = 0;
            zgc->ScrollMaxX = stopTime;
            //Desactiva la opción IsAutoScrollRange
            zgc->IsAutoScrollRange = false;
            //Habilita zoom y expandir horizontalmente
            zgc->IsEnableHPan = true;
            zgc->IsEnableHZoom = true;
            //Deshabilita zoom y expandir verticalmente
            zgc->IsEnableVPan = true;
            zgc->IsEnableVZoom = true;
            //Mueve rango del eje Y arriba o abajo para correponder a los datos
            zgc->GraphPane->IsBoundedRanges = true;
        }
    //Configuración eje x
    private: Void VariableEjeX(ZedGraphControl^  zg1)
        {
            GraphPane^ myPane = zg1->GraphPane;
            if(bEjexTiempo == false)
            {
               myPane->XAxis->Title->Text = "Número de Muestras";
               myPane->XAxis->Title->FontSpec->Size = 25.0F;
              //Modificar propiedades del Graphpane2
               myPane->XAxis->Scale->MinorStep = 1;
               myPane->XAxis->Scale->MajorStep = 10;
               myPane->XAxis->Scale->Min = 0;
               myPane->XAxis->Scale->Max = 10;
               zg1->Invalidate();
            }
            else // si ejex es tiempo
            {
               //Fija los títulos
               myPane->XAxis->Title->Text = "Tiempo (s)";
               myPane->XAxis->Title->FontSpec->Size = 25.0F;
               //Modificar propiedades del GraphPane2
               myPane->XAxis->Scale->MinorStep = 0.5;
               myPane->XAxis->Scale->MajorStep = 5;
               myPane->XAxis->Scale->Min = 0;
               myPane->XAxis->Scale->Max = 5;
               zg1->Invalidate();
            }
        }
    //Configuración eje y
    private: Void VariableEjeY(ZedGraphControl^  zg1)
        {
            GraphPane^ myPane = zg1->GraphPane;
            if(bEjeyAceleración == false)
            {
               myPane->YAxis->Title->Text = "Voltaje (V)";
               myPane->YAxis->Title->FontSpec->Size = 25.0F;
```

```
            /*myPane->YAxis->Scale->MinorStep = 0.2;
            myPane->YAxis->Scale->MajorStep = 0.5;
            myPane->YAxis->Scale->Min = 2;
            myPane->YAxis->Scale->Max = 8;*/
            //Modificar propiedades del Graphpane2
            zg1->Invalidate();
         }
         else // si ejey es aceleración
         {
            //Fija los títulos
             myPane->YAxis->Title->Text = "Aceleración (mg)";
            myPane->YAxis->Title->FontSpec->Size = 25.0F;
            //Modificar propiedades del GraphPane2
            zg1->Invalidate();
         }
      }
//Configuración mostrar puntos de las gráficas cuando se mueve el mouse encima de ellas
private: System::Boolean zedGraphControl1_MouseMoveEvent(ZedGraph::ZedGraphControl^  sender, System::Windows::Forms::MouseEventArgs^  e)
      {
         //Guarda ubicación del mouse
         PointF mouse  = System::Drawing::PointF(e->X, e->Y);
         //Encuentra el tramo de la gráfica que contiene la ubicación del mouse
         GraphPane^ pane = sender->MasterPane->FindChartRect(mouse);
         //Si el pane no es nulo, se tiene una ubicación válida. De lo contrario el mouse no está dentro de la gráfica
         if ( pane != nullptr )
         {
            double x, y;
            //Convierte la ubicación del mouse a valores de escala de X y Y
            pane->ReverseTransform(mouse, x, y);
            //Escribe los valores en el statuslabel
            toolStripStatusXY->Text = String::Concat("(X: ", x.ToString("f2"),", ", "Y: ",y.ToString("f2"),")");
         }
         else
            //Si no hay datos válidos, entonces limpia el statuslabel
            toolStripStatusXY->Text = String::Empty;

         //Retorna false para indicar que no se ha procesado el MouseMoveEvent
         //ZedGraphControl debe ir adelante y manejarlo
         return false;
      }

private: System::Boolean zedGraphControl2_MouseMoveEvent(ZedGraph::ZedGraphControl^  sender, System::Windows::Forms::MouseEventArgs^  e)
      {
         //Guarda ubicación del mouse
         PointF mouse  = System::Drawing::PointF(e->X, e->Y);
         //Encuentra el tramo de la gráfica que contiene la ubicación del mouse
         GraphPane^ pane = sender->MasterPane->FindChartRect(mouse);
         //Si el pane no es nulo, se tiene una ubicación válida. De lo contrario el mouse no está dentro de la gráfica
         if ( pane != nullptr )
         {
            double x, y;
            //Convierte la ubicación del mouse a valores de escala de X y Y
            pane->ReverseTransform(mouse, x, y);
            //Escribe los valores en el statuslabel
            toolStripStatusXY->Text = String::Concat("(X: ", x.ToString("f2"),", ", "Y: ",y.ToString("f2"),")");
         }
         else
            //Si no hay datos válidos, entonces limpia el statuslabel
            toolStripStatusXY->Text = String::Empty;

         //Retorna false para indicar que no se ha procesado el MouseMoveEvent
         //ZedGraphControl debe ir adelante y manejarlo
         return false;
      }
private: System::Boolean zedGraphControl3_MouseMoveEvent(ZedGraph::ZedGraphControl^  sender, System::Windows::Forms::MouseEventArgs^  e)
      {
         //Guarda ubicación del mouse
         PointF mouse  = System::Drawing::PointF(e->X, e->Y);
         //Encuentra el tramo de la gráfica que contiene la ubicación del mouse
         GraphPane^ pane = sender->MasterPane->FindChartRect(mouse);
         //Si el pane no es nulo, se tiene una ubicación válida. De lo contrario el mouse no está dentro de la gráfica
         if ( pane != nullptr )
         {
            double x, y;
            //Convierte la ubicación del mouse a valores de escala de X y Y
            pane->ReverseTransform(mouse, x, y);
            //Escribe los valores en el statuslabel
            toolStripStatusXY->Text = String::Concat("(X: ", x.ToString("f2"),", ", "Y: ",y.ToString("f2"),")");
         }
         else
            //Si no hay datos válidos, entonces limpia el statuslabel
            toolStripStatusXY->Text = String::Empty;

         //Retorna false para indicar que no se ha procesado el MouseMoveEvent
         //ZedGraphControl debe ir adelante y manejarlo
         return false;
      }
```

```
private: System::Void zedGraphControl1_ScrollProgressEvent(ZedGraph::ZedGraphControl^  sender, System::Windows::Forms::ScrollBar^  scrollBar,
          ZedGraph::ZoomState^  oldState, ZedGraph::ZoomState^  newState)
       {
          //this->toolStripStatusScroll->Text = String::Concat("Scrolling (Xmax: ",sender->GraphPane->XAxis->Scale->Max.ToString(),")");
       }

/*****************************************************/
/***       Configuración menú Visualizar            ***/
/*****************************************************/
private: System::Void tiempoToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e)
       {
          //Habilita visualización eje x tiempo
          bEjexTiempo = true;
          tiempoToolStripMenuItem->Checked = true;
          númeroDeMuestrasToolStripMenuItem->Checked = false;
          DiseñoGráfica();
       }
private: System::Void númeroDeMuestrasToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e)
       {
          //Habilita visualización eje x número de muestras
          bEjexTiempo = false;
          tiempoToolStripMenuItem->Checked = false;
          númeroDeMuestrasToolStripMenuItem->Checked = true;
          DiseñoGráfica();
       }
private: System::Void aceleraciónToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e)
       {
          //Habilita visualización eje y aceleración
          bEjeyAceleración = true;
          aceleraciónToolStripMenuItem->Checked = true;
          voltajeToolStripMenuItem->Checked = false;
          DiseñoGráfica();
       }
private: System::Void voltajeToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e)
       {
          //Habilita visualización eje y voltaje
          bEjeyAceleración = false;
          aceleraciónToolStripMenuItem->Checked = false;
          voltajeToolStripMenuItem->Checked = true;
          DiseñoGráfica();
       }
private: System::Void canal1ToolStripMenuItem1_Click(System::Object^  sender, System::EventArgs^  e)
       {
          //Habilita el control gráfico del canal1
          zedGraphControl1->Visible = true;
          zedGraphControl2->Visible = false;
          zedGraphControl3->Visible = false;
          //Chequea la opción Canal1 del menú
          canal1ToolStripMenuItem1->Checked = true;
          canal2ToolStripMenuItem1->Checked = false;
          canal3ToolStripMenuItem1->Checked = false;
          todosLosCanalesToolStripMenuItem1->Checked = false;
          //LLama a la funcion que reubica la gráfica del canal1
          NuevaUbicación(zedGraphControl1);
          lblGraf->Text = "Gráfica de visualización canal 1";

       }
private: System::Void canal2ToolStripMenuItem1_Click(System::Object^  sender, System::EventArgs^  e)
       {
          //Habilita el control gráfico del canal2
          zedGraphControl1->Visible = false;
          zedGraphControl2->Visible = true;
          zedGraphControl3->Visible = false;
          //Chequea la opción Canal2 del menú
          canal2ToolStripMenuItem1->Checked = true;
          canal1ToolStripMenuItem1->Checked = false;
          canal3ToolStripMenuItem1->Checked = false;
          todosLosCanalesToolStripMenuItem1->Checked = false;
          //LLama a la función que reubica la gráfica del canal2
          NuevaUbicación(zedGraphControl2);
          lblGraf->Text = "Gráfica de visualización canal 2";
       }
private: System::Void canal3ToolStripMenuItem1_Click(System::Object^  sender, System::EventArgs^  e)
       {
          //Habilita el control gráfico del canal3
          zedGraphControl1->Visible = false;
          zedGraphControl2->Visible = false;
          zedGraphControl3->Visible = true;
          //Chequea la opción Canal3 del menú
          canal3ToolStripMenuItem1->Checked = true;
          canal1ToolStripMenuItem1->Checked = false;
          canal2ToolStripMenuItem1->Checked = false;
          todosLosCanalesToolStripMenuItem1->Checked = false;
          //LLama a la función que reubica la gráfica del canal3
          NuevaUbicación(zedGraphControl3);
          lblGraf->Text = "Gráfica de visualización canal 3";
       }
private: System::Void todosLosCanalesToolStripMenuItem1_Click(System::Object^  sender, System::EventArgs^  e)
```

```
       {
          //Habilita los tres controles gráficos: canal 1, 2 y 3
          zedGraphControl1->Visible = true;
          zedGraphControl2->Visible = true;
          zedGraphControl3->Visible = true;

          //Chequea la opción todosloscanales del menú
          canal1ToolStripMenuItem1->Checked = false;
          canal2ToolStripMenuItem1->Checked = false;
          canal3ToolStripMenuItem1->Checked = false;
          todosLosCanalesToolStripMenuItem1->Checked = true;

          //Ubica el panel1 en su posición inicial
          zedGraphControl1->Location = Point(353,30);
          NuevoSize(zedGraphControl1);

          //Ubica el panel2 en su posición inicial
          zedGraphControl2->Location = Point(353, 260);
          NuevoSize(zedGraphControl2);

          //Ubica el panel3 en su posición inicial
          zedGraphControl3->Location = Point(353, 490);
          NuevoSize(zedGraphControl3);

          lblGraf->Text = "Gráficas de visualización canales: 1, 2 y 3";
          lblGraf->Location = Point(736,8);
       }
private: Void NuevaUbicación(ZedGraphControl^ zgc)
       {
          lblGraf->Location = Point(780,115);
          //Ubica el control de la gráfica en una nueva posición
          //Define un nuevo tamaño para la gráfica
          zgc->Location =  Point(420,140);
          zgc->Size =  System::Drawing::Size(900,450);

          //Cambia el tamaño de las fuentes de la gráfica
          GraphPane^ myPane = zgc->GraphPane;
          myPane->Title->FontSpec->Size = 12.0F;
          myPane->XAxis->Title->FontSpec->Size = 12.0F;
          myPane->YAxis->Title->FontSpec->Size = 12.0F;
          myPane->XAxis->Scale->FontSpec->Size = 10.0F;
          myPane->YAxis->Scale->FontSpec->Size = 10.0F;
       }

/*****************************************************/
/***   Configuración groupBox Canales de Entrada    ***/
/*****************************************************/
private: Void ElegirCanal()
       {
          //Dependiendo del radiobutton escogido se fija el canal a adquirir
          //usando la variable intCCanal
          switch (intCCanal)
          {
          case 1:
             char * p;
             adq->can = true;
             p = adq->canales;
              *p = 'D'; p++; *p = 'e'; p++; *p = 'v'; p++; *p = '1'; p++; *p = '/';
             p++; *p = 'a'; p++; *p = 'i'; p++; *p = '0'; p++; *p = '\0';
             break;
          case 2:
             char * p2;
             adq->can = true;
             p2 = adq->canales;
              *p2 = 'D'; p2++; *p2 = 'e'; p2++; *p2 = 'v'; p2++; *p2 = '1'; p2++; *p2 = '/';
             p2++; *p2 = 'a'; p2++; *p2 = 'i'; p2++; *p2 = '1'; p2++; *p2= '\0';
             break;
          case 3:
             char * p3;
             adq->can = true;
             p3 = adq->canales;
              *p3 = 'D'; p3++; *p3 = 'e'; p3++; *p3 = 'v'; p3++; *p3 = '1'; p3++; *p3 = '/';
             p3++; *p3 = 'a'; p3++; *p3 = 'i'; p3++; *p3 = '2'; p3++; *p3 = '\0';
             break;
          case 4:
             char * p4;
             adq->can = false;
             p4 = adq->canales;
             *p4 = 'D'; p4++; *p4 = 'e'; p4++; *p4 = 'v'; p4++; *p4 = '1'; p4++; *p4 = '/';
             p4++; *p4 = 'a'; p4++; *p4 = 'i'; p4++; *p4 = '0'; p4++; *p4 = ','; p4++;
             *p4 = 'D'; p4++; *p4 = 'e'; p4++; *p4 = 'v'; p4++; *p4 = '1'; p4++; *p4 = '/';
             p4++; *p4 = 'a'; p4++; *p4 = 'i'; p4++; *p4 = '1'; p4++; *p4 = ','; p4++;
             *p4 = 'D'; p4++; *p4 = 'e'; p4++; *p4 = 'v'; p4++; *p4 = '1'; p4++; *p4 = '/';
             p4++; *p4 = 'a'; p4++; *p4 = 'i'; p4++; *p4 = '2'; p4++; *p4 = '\0';
             break;
          }
       }
private: System::Void rdbCanal1_CheckedChanged(System::Object^  sender, System::EventArgs^  e)
```

```
       {
          intCCanal = 1;
          //Deshabilita los controles gráficos de los canales 2 y 3
          zedGraphControl2->Visible = true;
          zedGraphControl2->Visible = false;
          zedGraphControl3->Visible = false;

          //Ubica el panel1 en posición central
          NuevaUbicación(zedGraphControl1);
          lblGraf->Text = "Gráfica de visualización canal 1";
       }
private: System::Void rdbCanal2_CheckedChanged(System::Object^  sender, System::EventArgs^  e)
       {
          intCCanal = 2;
          //Deshabilita los controles gráficos de los canales 2 y 3
          zedGraphControl1->Visible = false;
          zedGraphControl2->Visible = true;
          zedGraphControl3->Visible = false;

          //Ubica el panel2 en posición central
          NuevaUbicación(zedGraphControl2);
          lblGraf->Text = "Gráfica de visualización canal 2";
       }
private: System::Void rdbCanal3_CheckedChanged(System::Object^  sender, System::EventArgs^  e)
       {
          intCCanal = 3;
          //Deshabilita los controles gráficos de los canales 2 y 3
          zedGraphControl1->Visible = false;
          zedGraphControl2->Visible = false;
          zedGraphControl3->Visible = true;

          //Ubica el panel2 en posición central
          NuevaUbicación(zedGraphControl3);
          lblGraf->Text = "Gráfica de visualización canal 3";
       }
private: System::Void chkCanales_CheckedChanged(System::Object^  sender, System::EventArgs^  e)
       {
          if(chkCanales->Checked == true)
          {
             //Deshabilita los botones de los canales
             rdbCanal1->Enabled = false;
             rdbCanal2->Enabled = false;
             rdbCanal3->Enabled = false;
             rdbCanal1->Checked = false;
             rdbCanal2->Checked = false;
             rdbCanal3->Checked = false;

             //Habilita los tres controles gráficos: canal 1, 2 y 3
             zedGraphControl1->Visible = true;
             zedGraphControl2->Visible = true;
             zedGraphControl3->Visible = true;

             //Chequea la opción todosloscanales del menú
             canal1ToolStripMenuItem1->Checked = false;
             canal2ToolStripMenuItem1->Checked = false;
             canal3ToolStripMenuItem1->Checked = false;
             todosLosCanalesToolStripMenuItem1->Checked = true;

             //Ubica el panel1 en su posición inicial
             zedGraphControl1->Location = Point(353,30);
             NuevoSize(zedGraphControl1);

             //Ubica el panel2 en su posición inicial
             zedGraphControl2->Location = Point(353, 260);
             NuevoSize(zedGraphControl2);

             //Ubica el panel3 en su posición inicial
             zedGraphControl3->Location = Point(353, 490);
             NuevoSize(zedGraphControl3);

             intCCanal = 4;

             lblGraf->Text = "Gráficas de visualización canales: 1, 2 y 3";
             lblGraf->Location = Point(736,8);
          }
          else
          {     //Habilita los botones para escoger uno de los canales
                //Chequea el canal1 por defecto
                rdbCanal1->Enabled = true;
                rdbCanal2->Enabled = true;
                rdbCanal3->Enabled = true;
                rdbCanal1->Checked = true;
          }
       }
private: Void NuevoSize(ZedGraphControl^ zgc)
       {
          zgc->Size =  System::Drawing::Size(990,230);
          GraphPane^ myPane = zgc->GraphPane;
```

```
            myPane->Title->FontSpec->Size = 25.0F;
            myPane->XAxis->Title->FontSpec->Size = 25.0F;
            myPane->YAxis->Title->FontSpec->Size = 25.0F;
            myPane->XAxis->Scale->FontSpec->Size = 22.0F;
            myPane->YAxis->Scale->FontSpec->Size = 22.0F;
            zgc->Refresh();
        }

/*****************************************************/
/***  Configuración groupBox Parámetros de Tiempo   ***/
/*****************************************************/
private: Void ConfigurarParamTiempos()
        {
            //Asignación de valores a las variables
            //Tasa de muestreo (Hz)
            flTasaMuestreo = (float64) this->numSampleRate->Value;
            adq->sampleRate = flTasaMuestreo;
            //PointsToRead = tasa de muestreo
            adq->pointsToRead = (int)flTasaMuestreo;
            //Duración del muestreo
            intDuraciónMuestreo = (int) this->numDuraciónMuestreo->Value;

            if(rdbMinutos->Checked == true)
            {
               stopTime = intDuraciónMuestreo * 60;
            }
            if(rdbSegundos->Checked == true)
            {
               stopTime = intDuraciónMuestreo;
            }

            //Si el eje x es numero de muestras
            if(bEjexTiempo == false)
            {
               //Define el máximo del eje x
               valorMax = 3 * adq->pointsToRead;
               //Canal1
               zedGraphControl1->GraphPane->XAxis->Scale->Max = valorMax;
               zedGraphControl1->GraphPane->XAxis->Scale->MajorStep = adq->pointsToRead;
               zedGraphControl1->GraphPane->XAxis->Scale->MinorStep = (adq->pointsToRead / 10);
                //Canal2
               zedGraphControl2->GraphPane->XAxis->Scale->Max = valorMax;
               zedGraphControl2->GraphPane->XAxis->Scale->MajorStep = adq->pointsToRead;
               zedGraphControl2->GraphPane->XAxis->Scale->MinorStep = (adq->pointsToRead / 10);
               //Canal3
               zedGraphControl3->GraphPane->XAxis->Scale->Max = valorMax;
               zedGraphControl3->GraphPane->XAxis->Scale->MajorStep = adq->pointsToRead;
               zedGraphControl3->GraphPane->XAxis->Scale->MinorStep = (adq->pointsToRead / 10);
            }

            //Si el eje x es tiempo
            else
            {
               //Define el máximo del eje x
               valorMax = 5;
               //Canal1
               zedGraphControl1->GraphPane->XAxis->Scale->Max = 5;
               zedGraphControl1->GraphPane->XAxis->Scale->Min = 0;
               zedGraphControl1->GraphPane->XAxis->Scale->MajorStep = 5;
               zedGraphControl1->GraphPane->XAxis->Scale->MinorStep = 1;
               zedGraphControl1->Invalidate();
               //Canal2
               zedGraphControl2->GraphPane->XAxis->Scale->Max = 5;
               zedGraphControl2->GraphPane->XAxis->Scale->MajorStep = 5;
               zedGraphControl2->GraphPane->XAxis->Scale->MinorStep = 0.5;
               zedGraphControl2->Invalidate();
               //Canal3
               zedGraphControl3->GraphPane->XAxis->Scale->Max = 5;
               zedGraphControl3->GraphPane->XAxis->Scale->MajorStep = 5;
               zedGraphControl3->GraphPane->XAxis->Scale->MinorStep = 0.5;
               zedGraphControl3->Invalidate();
            }
        }
private: System::Void rdbMinutos_CheckedChanged(System::Object^  sender, System::EventArgs^  e)
        {
            lblDuraciónMuestreo->Text = "Lapso de tiempo (min)";
            numDuraciónMuestreo->Maximum = 30;
            numDuraciónMuestreo->Minimum = 1;
        }
private: System::Void rdbSegundos_CheckedChanged(System::Object^  sender, System::EventArgs^  e)
        {
            lblDuraciónMuestreo->Text = "Lapso de tiempo (s)";
            numDuraciónMuestreo->Maximum = 60;
            numDuraciónMuestreo->Minimum = 1;
        }

/*****************************************************/
/*  Configuración Botones Inicio y Parar Adquisición  */
```

```
/*****************************************************/
private: System::Void btnIniciar_Click(System::Object^  sender, System::EventArgs^  e)
       {
          //Variable para controlar  tiempo
          time_t  a;
          int chan = 0;
          int numero = 0;
          String^ nombreArchivo;

          //Configuración del toolStatusStrip
          toolLabel1->Text = "Configurando Parámetros...";

          //Configuración de los controles
          //Se deshabilitan o habilitan mientras esten adquiriendo, se vuelven a habilitar
          //con el botón Parar o al terminar la adquisición durante el tiempo estimado
          grbCanales->Enabled = false;
          grbTiempo->Enabled = false;
          btnIniciar->Enabled = false;
          visualizarToolStripMenuItem->Enabled = false;
          grbEventos->Enabled = true;
          btnIniciar->Enabled = false;
          btnParar->Enabled = true;
          chkGuardar->Enabled = false;

          //Llama a la función para saber los parámetros de tiempo
          ConfigurarParamTiempos();

          if(chkGuardar->Checked == true)
          {
             //Abre el cuadro de diálogo Guardar como
Guardar:     if(this->saveFileDialog1->ShowDialog() == Windows::Forms::DialogResult::OK)
             {
                //Guarda el nombre de archivo que el usuario digite
                nombreArchivo = gcnew String(saveFileDialog1->FileName);
                fileDatos = gcnew FileInfo(nombreArchivo);
             }
             else
             {
                System::Windows::Forms::DialogResult pregunta;
                pregunta = MessageBox::Show("Cancelar adquisición?","Confirmar cancelar",MessageBoxButtons::YesNo,
                   MessageBoxIcon::Question);
                if(pregunta == ::DialogResult::Yes)
                {
                   toolLabel1->Text = "Adquisición Detenida";
                   //Habilita nuevamente los botones para configurar adquisición
                   HabilitarBotones();
                   toolLabel1->Text = "Listo para adquirir";
                   goto Etiqueta;
                }
                else
                   goto Guardar;

             }
          }
          //Empieza el proceso de adquisición
          //Reinicia el timer y los contadores de incremento para leer los arrays
          intCounterCanal1 = 0;
          intCounterCanal2 = 0;
          intCounterCanal3 = 0;
          timerGraficar->Start();
          intIncremento1 = 0;
          intIncremento2 = 0;
          intIncremento3 = 0;

          //Crea el task
          adq->CreateTask();
          ComprobarError();
          if(hError == true)
             goto Etiqueta;
          //Llama a la función para saber que canal ha escogido
          ElegirCanal();
          //Crea el canal de entrada
          adq->CreateAIVoltageChan();
          ComprobarError();
          if(hError == true)
             goto Etiqueta;
          //Llama a la función para saber los parámetros de tiempo
          ConfigurarParamTiempos();
          //Configura el clock de la tarjeta
          adq->CfgSampClkTiming();
          ComprobarError();
          if(hError == true)
             goto Etiqueta;
          //Configura el buffer
          adq->CfgInputBuffer();
          ComprobarError();
          if(hError == true)
             goto Etiqueta;
```

```
//Cambia de estado al task para empezar a adquirir
adq->StartTask();
ComprobarError();
if(hError == true)
   goto Etiqueta;

DateTime TiempoInicio = DateTime::Now;
toolLabel1->Text = "Adquiriendo...";

//Configuración groupBox Evento
txtSitio->Text = varSitio;
txtInicio->Text = TiempoInicio.ToString("T");
txtFecha->Text = TiempoInicio.ToString("D");
txtArchivo->Text = nombreArchivo;

//Comienza a leer
a = time(NULL);
while(time(NULL)< a + stopTime)
{
   adq->ReadAnalog();
   if(adq->error != 0)
   {
      adq->StopTask();
      adq->ClearTask();
      goto Etiqueta;
   }

   switch (intCCanal)
   {
   case 1:
      for(int i = 0; i < adq->pointsRead; i++)
      datosCanal1->SetValue(adq->datos[i],i);
      break;
   case 2:
      for(int i = 0; i < adq->pointsRead; i++)
      datosCanal2->SetValue(adq->datos[i],i);
      break;
   case 3:
      for(int i = 0; i < adq->pointsRead; i++)
      datosCanal3->SetValue(adq->datos[i],i);
      break;
   case 4:
      for(int i = 0; i < adq->pointsRead; i++)
      {
      datosCanal1->SetValue(adq->data[i*3+0],i);
      datosCanal2->SetValue(adq->data[i*3+1],i);
      datosCanal3->SetValue(adq->data[i*3+2],i);
      }
      break;
   }
    //Aplicar filtro notch a los datos
    FiltroNotch(datosCanal1);
    FiltroNotch2(datosCanal2);
    FiltroNotch3(datosCanal3);

    //Crea el archivo donde se va a guardar los datos
    if(chkGuardar->Checked == true)
    {
      try
      {
         fileDatos = gcnew FileInfo(nombreArchivo);
         //Si ya existe adiciona los nuevos datos al final del archivo
         if(fileDatos->Exists == true)
           strDatos = fileDatos->AppendText();
         //Si no existe crea un archivo de texto nuevo
         else
         {
           strDatos = fileDatos->CreateText();
           //Encabezado del archivo
           DateTime curTime = DateTime::Now;
           strDatos->WriteLine("Universidad de Nariño");
           strDatos->WriteLine("Facultad de Ingeniería");
           strDatos->WriteLine("Adquisición y Registro de Microtrepidaciones");
           strDatos->Write("Proyecto: ");
           strDatos->WriteLine(varProyecto);
           strDatos->Write("Operador: ");
           strDatos->WriteLine(varOperador);
           strDatos->Write("Sitio: ");
           strDatos->WriteLine(varSitio);
           strDatos->WriteLine("Separador: Doble espacio");
           strDatos->Write("Fecha: ");
           strDatos->WriteLine(curTime.ToString("D"));
           strDatos->Write("Hora Inicio: ");
           strDatos->WriteLine(curTime.ToString("T"));
           strDatos->WriteLine("Lapso de tiempo (s): ");
           strDatos->WriteLine(stopTime.ToString());
           if(intCCanal > 3)
               chan = 3;
```

```
                    else
                        chan = 1;
                    strDatos->WriteLine("Canales: ");
                    strDatos->WriteLine(chan.ToString());
                    strDatos->WriteLine("Frecuencia de muestreo (Hz):");
                    strDatos->WriteLine(flTasaMuestreo.ToString());
                    strDatos->WriteLine("***** Fin Encabezado *****");
                  }
                  //Escribe los datos que se van adquiriendo en el archivo
                  for(int k = 0; k< adq->pointsRead; k++)
                  {
                     if(adq->can == true)
                     {
                        numero++;
                        strDatos->WriteLine("{0}  {1:g}",numero,adq->datos[k]);
                     }
                     else
                     {
                        numero++;
                        //strDatos->WriteLine("{0:g}  {1:g} ",datosCanal1[k],datosCanal2[k]);
                        strDatos->WriteLine("{0}  {1:g}  {2:g}  {3:g} ",numero,datosCanal1[k],
                           datosCanal2[k],datosCanal3[k]);
                        //strDatos->WriteLine("{0}  {1:g}  {2:g}  {3:g} ",numero,adq->data[k*3+0],
                        //adq->data[k*3+1],adq->data[k*3+2]);
                        //strDatos->WriteLine("{0} {1:g} {2:g} ", adq->data[i*3+0],adq->data[i*3+1],
                        //adq->data[i*3+2]);
                     }
                  }
                  //Si el usuario escogio aceleración para el eje y, hace la conversion de voltaje a aceleración
               if(bEjeyAceleración == true)
               {
                  for(int k = 0; k< adq->pointsRead; k++)
                  {
                     datosCanal1[k] = ((datosCanal1[k] - 6.658) / (2.7953*0.498))*1000;
                     datosCanal2[k] = ((datosCanal2[k] - 6.718) / (2.8371*0.499))*1000;
                     datosCanal3[k] = ((datosCanal3[k] - 8.044) / (2.7992*0.5))*1000;
                  }
               }
                 }

                 __finally
                 {
                    //Cierra el objeto StreamWriter para liberar recursos
                    strDatos->Flush();
                    strDatos->Close();

                 }
              }
           }
           //Detiene el task
           adq->StopTask();
           adq->ClearTask();
           /*ComprobarError();*/

Etiqueta:   toolLabel1->Text = "Listo para adquirir";
        }
private: System::Void btnParar_Click(System::Object^  sender, System::EventArgs^  e)
        {
           toolLabel1->Text = "Adquisición Detenida";
           //Habilita nuevamente los botones para configurar adquisición
           HabilitarBotones();
           //Detiene el timer de las gráficas
           timerGraficar->Stop();
           timerGraficar->Enabled = false;
           //Detiene el task
           adq->StopTask();
           adq->ClearTask();
           toolLabel1->Text = "Listo para Adquirir";
        }

/*****   Subrutinas   ******/
private: Void ComprobarError()
        {
           if(adq->error != 0)
           {
              hError = true;
              String^ strError = nullptr;
              char * chError;
              chError = new char [10000];
              System::Windows::Forms::DialogResult respuesta;

              for(int i = 0; i < 2048; i++)
              {
                chError[i] = adq->errBuff[i];
              }

              strError = gcnew String (chError,0,10000);
              respuesta = MessageBox::Show(String::Concat("Revisar conexión de la tarjeta al puerto USB y demás cableado. DAQmxBase Error: "
```

```
               ,strError),
               String::Concat("Error ", adq->error),MessageBoxButtons::OK,
               MessageBoxIcon::Error);

            toolLabel1->Text = "Adquisición Detenida";
            //Habilita nuevamente los botones para configurar adquisición
            HabilitarBotones();
            //Detiene el timer donde se programa la gráfica
            timerGraficar->Stop();
            //Detiene el task
            adq->StopTask();
            adq->ClearTask();
         }
         else
            hError = false;
      }
private: Void HabilitarBotones()
      {
         //Habilitan los controles para empezar de nuevo la adquisición
         grbCanales->Enabled = true;
         grbTiempo->Enabled = true;
         btnIniciar->Enabled = true;
         visualizarToolStripMenuItem->Enabled = true;
         grbEventos->Enabled = false;
         btnIniciar->Enabled = true;
         btnParar->Enabled = false;
         chkGuardar->Enabled = true;
      }
private: Void FiltroNotch(array<double>^ df)
{
   #define NZEROS 12
   #define NPOLES 12
   #define GAIN   1.686639413e+00      //Ganancia fs = 200Hz
   #define GAIN1  1.518327605e+00      //Ganancia fs = 250Hz
   #define GAIN2  1.415872176e+00      //Ganancia fs = 300Hz

   static double xv[NZEROS+1], yv[NPOLES+1];
   array<double>^ datosfiltro = gcnew array<double> (10000);

   for(int k = 0; k< adq->pointsRead; k++)
   {
      xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4]; xv[4] = xv[5]; xv[5] = xv[6]; xv[6] = xv[7]; xv[7] = xv[8];
      xv[8] = xv[9]; xv[9] = xv[10]; xv[10] = xv[11]; xv[11] = xv[12];
      xv[12] = df[k] / GAIN;
      yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4]; yv[4] = yv[5]; yv[5] = yv[6]; yv[6] = yv[7]; yv[7] = yv[8];
      yv[8] = yv[9]; yv[9] = yv[10]; yv[10] = yv[11]; yv[11] = yv[12];
      yv[12] =   (xv[0] + xv[12]) +   3.7168023503 * (xv[1] + xv[11]) +  11.7560915460 * (xv[2] + xv[10])
         +  23.3382905480 * (xv[3] + xv[9]) +  40.2332055110 * (xv[4] + xv[8]) +  51.9781811740 * (xv[5] + xv[7])
         +  59.0107358460 * xv[6]
         + ( -0.3515244284 * yv[0]) + ( -1.4198733686 * yv[1])
         + ( -4.8681782816 * yv[2]) + (-10.5081362870 * yv[3])
         + (-19.6863212650 * yv[4]) + (-27.7090035860 * yv[5])
         + (-34.2724017790 * yv[6]) + (-32.9724037690 * yv[7])
         + (-27.8755622870 * yv[8]) + (-17.7130256690 * yv[9])
         + ( -9.7673534271 * yv[10]) + ( -3.3944163715 * yv[11]);
      datosfiltro[k] = yv[12];
   }
      for(int i = 0; i< 10000; i++)
         datosCanal1[i] = datosfiltro[i];

}
private: Void FiltroNotch2(array<double>^ df)
{
   #define NZEROS 12
   #define NPOLES 12
   #define GAIN   1.686639413e+00      //Ganancia fs = 200Hz
   #define GAIN1  1.518327605e+00      //Ganancia fs = 250Hz
   #define GAIN2  1.415872176e+00      //Ganancia fs = 300Hz

   static double xv[NZEROS+1], yv[NPOLES+1];
   array<double>^ datosfiltro1 = gcnew array<double> (10000);

   for(int k = 0; k< adq->pointsRead; k++)
   {
      xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4]; xv[4] = xv[5]; xv[5] = xv[6]; xv[6] = xv[7]; xv[7] = xv[8];
      xv[8] = xv[9]; xv[9] = xv[10]; xv[10] = xv[11]; xv[11] = xv[12];
      xv[12] = df[k] / GAIN;
      yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4]; yv[4] = yv[5]; yv[5] = yv[6]; yv[6] = yv[7]; yv[7] = yv[8];
      yv[8] = yv[9]; yv[9] = yv[10]; yv[10] = yv[11]; yv[11] = yv[12];
      yv[12] =   (xv[0] + xv[12]) +   3.7168023503 * (xv[1] + xv[11]) +  11.7560915460 * (xv[2] + xv[10])
         +  23.3382905480 * (xv[3] + xv[9]) +  40.2332055110 * (xv[4] + xv[8]) +  51.9781811740 * (xv[5] + xv[7])
         +  59.0107358460 * xv[6]
         + ( -0.3515244284 * yv[0]) + ( -1.4198733686 * yv[1])
         + ( -4.8681782816 * yv[2]) + (-10.5081362870 * yv[3])
         + (-19.6863212650 * yv[4]) + (-27.7090035860 * yv[5])
         + (-34.2724017790 * yv[6]) + (-32.9724037690 * yv[7])
         + (-27.8755622870 * yv[8]) + (-17.7130256690 * yv[9])
         + ( -9.7673534271 * yv[10]) + ( -3.3944163715 * yv[11]);
```

```
      datosfiltro1[k] = yv[12];
   }

   for(int i = 0; i< 10000; i++)
      datosCanal2[i] = datosfiltro1[i];

}
private: Void FiltroNotch3(array<double>^ df)
{
   #define NZEROS 12
   #define NPOLES 12
   #define GAIN   1.686639413e+00      //Ganancia fs = 200Hz
   #define GAIN1  1.518327605e+00      //Ganancia fs = 250Hz
   #define GAIN2  1.415872176e+00      //Ganancia fs = 300Hz

   static double xv[NZEROS+1], yv[NPOLES+1];
   array<double>^ datosfiltro2 = gcnew array<double> (10000);

   for(int k = 0; k< adq->pointsRead; k++)
   {
      xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4]; xv[4] = xv[5]; xv[5] = xv[6]; xv[6] = xv[7]; xv[7] = xv[8];
      xv[8] = xv[9]; xv[9] = xv[10]; xv[10] = xv[11]; xv[11] = xv[12];
      xv[12] = df[k] / GAIN;
      yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4]; yv[4] = yv[5]; yv[5] = yv[6]; yv[6] = yv[7]; yv[7] = yv[8];
      yv[8] = yv[9]; yv[9] = yv[10]; yv[10] = yv[11]; yv[11] = yv[12];
      yv[12] =   (xv[0] + xv[12]) +   3.7168023503 * (xv[1] + xv[11]) +  11.7560915460 * (xv[2] + xv[10])
         +  23.3382905480 * (xv[3] + xv[9]) +  40.2332055110 * (xv[4] + xv[8]) +  51.9781811740 * (xv[5] + xv[7])
         +  59.0107358460 * xv[6]
         + ( -0.3515244284 * yv[0]) + ( -1.4198733686 * yv[1])
         + ( -4.8681782816 * yv[2]) + (-10.5081362870 * yv[3])
         + (-19.6863212650 * yv[4]) + (-27.7090035860 * yv[5])
         + (-34.2724017790 * yv[6]) + (-32.9724037690 * yv[7])
         + (-27.8755622870 * yv[8]) + (-17.7130256690 * yv[9])
         + ( -9.7673534271 * yv[10]) + ( -3.3944163715 * yv[11]);
      datosfiltro2[k] = yv[12];
   }
   for(int i = 0; i< 10000; i++)
      datosCanal3[i] = datosfiltro2[i];
}

/*****************************************************/
/******           Configuración timer           ******/
/*****************************************************/
private: System::Void timerGraficar_Tick(System::Object^  sender, System::EventArgs^  e)
       {
          int n = 0; // número de muestras
          double t1 = 0, t2= 0, t3= 0;

          //Curva Canal 1
          // Make sure that the curvelist has at least one curve
          if(zedGraphControl1->GraphPane->CurveList->Count <= 0)
             return;

          // Get the first CurveItem in the graph
          LineItem^ curva1 = (LineItem^)zedGraphControl1->GraphPane->CurveList[0];
          if(curva1)
          {
             // Get the PointPairList
             IPointListEdit^ list = (IPointListEdit^) curva1->Points;
             if ( list == nullptr )
                return;

             if(list)
             {
                if(intCounterCanal1 == 0)
                   //Limpia la lista de datos
                   list->Clear();

                //Se asegura que ha empezado a adquirir
                if(Array::TrueForAll(datosCanal1, gcnew Predicate<double>(IsZero)) == false)
                {
                   time1 = gcnew array<double> (flTasaMuestreo*stopTime);
                   for(int i = 0; i< time1->Length; i++)
                   {
                     time1->SetValue((t1),i);
                     t1 += 1/flTasaMuestreo;
                   }

                   if(bEjexTiempo == false)
                   {
                      double time = (Environment::TickCount - intTickStart)/1000;
                      for(int j = 0, n = intIncremento1; j < adq->pointsRead, n < (int)flTasaMuestreo + intIncremento1; j++, n++)
                      {
                         list->Add(n,datosCanal1[j]);
                         if(n > zedGraphControl1->GraphPane->XAxis->Scale->Max - zedGraphControl1->GraphPane->XAxis->Scale->MajorStep )
                         {
                            zedGraphControl1->GraphPane->XAxis->Scale->Max = n;
                            zedGraphControl1->GraphPane->XAxis->Scale->Min = zedGraphControl1->GraphPane->XAxis->Scale->Max - valorMax;
```

```
                }
            }
         }
         else
         {
            for(int j = 0, n = intIncremento1; j < adq->pointsRead, n < (int)flTasaMuestreo + intIncremento1; j++, n++)
            {
               list->Add(time1[n],datosCanal1[j]);
               if(time1[n] > zedGraphControl1->GraphPane->XAxis->Scale->Max - zedGraphControl1->GraphPane->XAxis->Scale->MajorStep )
               {
                  zedGraphControl1->GraphPane->XAxis->Scale->Max = time1[n];
                  zedGraphControl1->GraphPane->XAxis->Scale->Min = zedGraphControl1->GraphPane->XAxis->Scale->Max - 5;
               }
            }
         }
         zedGraphControl1->AxisChange();
         zedGraphControl1->Invalidate();
         zedGraphControl1->Refresh();
         intCounterCanal1++;

         intIncremento1 += (int)flTasaMuestreo;

         if(intIncremento1 >= (flTasaMuestreo * stopTime) )
         {
            timerGraficar->Enabled = false;
            HabilitarBotones();

         }
         txtTranscurrido->Text = intCounterCanal1.ToString();
      }
   }
}
//Curva canal2
// Make sure that the curvelist has at least one curve
if ( zedGraphControl2->GraphPane->CurveList->Count <= 0 )
   return;

// Get the first CurveItem in the graph
LineItem^ curva2 = (LineItem^)zedGraphControl2->GraphPane->CurveList[0];
if(curva2 == nullptr)
   return;
if(curva2)
{
   // Get the PointPairList
   IPointListEdit^ list2 = (IPointListEdit^) curva2->Points;
   // If this is null, it means the reference at curve->Points does not
   // support IPointListEdit, so we won't be able to modify it
   if ( list2 == nullptr )
      return;
   if(list2)
   {
      if(intCounterCanal2 == 0)
         //Limpia la lista de datos
         list2->Clear();
      //Se asegura que ha empezado a adquirir
      if(Array::TrueForAll(datosCanal2, gcnew Predicate<double>(IsZero)) == false)
      {
         time2 = gcnew array<double> (flTasaMuestreo*stopTime);
         for(int i = 0; i<time2->Length; i++)
         {
            time2->SetValue((t2),i);
            t2 += 1/flTasaMuestreo;
         }

         if(bEjexTiempo == false)
         {
            for(int j = 0, n = intIncremento2; j < adq->pointsRead, n < (int)flTasaMuestreo + intIncremento2; j++, n++)
            {
               list2->Add(n,datosCanal2[j]);
               if(n > zedGraphControl2->GraphPane->XAxis->Scale->Max - zedGraphControl2->GraphPane->XAxis->Scale->MajorStep )
               {
                  zedGraphControl2->GraphPane->XAxis->Scale->Max = n;
                  zedGraphControl2->GraphPane->XAxis->Scale->Min = zedGraphControl2->GraphPane->XAxis->Scale->Max - valorMax;
               }
            }
         }
         else
         {
            for(int j = 0, n = intIncremento2; j < adq->pointsRead, n < (int)flTasaMuestreo + intIncremento2; j++, n++)
            {
               list2->Add(time2[n], datosCanal2[j]);
               if(time2[n] > zedGraphControl2->GraphPane->XAxis->Scale->Max - zedGraphControl2->GraphPane->XAxis->Scale->MajorStep )
               {
                  zedGraphControl2->GraphPane->XAxis->Scale->Max = time2[n];
                  zedGraphControl2->GraphPane->XAxis->Scale->Min = zedGraphControl2->GraphPane->XAxis->Scale->Max - 5;
               }
            }
         }
```

```
                zedGraphControl2->AxisChange();
                zedGraphControl2->Invalidate();
                intCounterCanal2++;
                intIncremento2 += (int)flTasaMuestreo;

                if(intIncremento2 >= (flTasaMuestreo * stopTime) )
                {
                   timerGraficar->Enabled = false;
                   HabilitarBotones();
                }
                txtTranscurrido->Text = intCounterCanal2.ToString();
             }
          }
       }
       //Curva canal3
       // Make sure that the curvelist has at least one curve
       if ( zedGraphControl3->GraphPane->CurveList->Count <= 0 )
          return;
       // Get the first CurveItem in the graph
       LineItem^ curva3 = (LineItem^)zedGraphControl3->GraphPane->CurveList[0];
       if(curva3 == nullptr)
          return;
       if(curva3)
       {
          // Get the PointPairList
          IPointListEdit^ list3 = (IPointListEdit^) curva3->Points;
          // If this is null, it means the reference at curve->Points does not
          // support IPointListEdit, so we won't be able to modify it
          if ( list3 == nullptr )
             return;
          if(list3)
          {
             if(intCounterCanal3 == 0)
                //Limpia la lista de datos
                list3->Clear();

             //Se asegura que ha empezado a adquirir
             if(Array::TrueForAll(datosCanal3, gcnew Predicate<double>(IsZero)) == false)
             {
                time3 = gcnew array<double> (flTasaMuestreo * stopTime);
                for(int i = 0; i < time3->Length; i++)
                {
                   time3->SetValue((t3), i);
                   t3 += 1/flTasaMuestreo;
                }
                if(bEjexTiempo == false)
                {
                   for(int j = 0, n = intIncremento3; j < adq->pointsRead, n < (int)flTasaMuestreo + intIncremento3; j++, n++)
                   {
                      list3->Add(n,datosCanal3[j]);
                      if(n > zedGraphControl3->GraphPane->XAxis->Scale->Max - zedGraphControl3->GraphPane->XAxis->Scale->MajorStep )
                      {
                        zedGraphControl3->GraphPane->XAxis->Scale->Max = n;
                        zedGraphControl3->GraphPane->XAxis->Scale->Min = zedGraphControl3->GraphPane->XAxis->Scale->Max - valorMax;
                      }
                   }
                }
                else
                {
                   for(int j = 0, n = intIncremento3; j < adq->pointsRead, n < (int)flTasaMuestreo + intIncremento3; j++, n++)
                   {
                      list3->Add(time3[n], datosCanal3[j]);
                      if(time3[n] > zedGraphControl3->GraphPane->XAxis->Scale->Max - zedGraphControl3->GraphPane->XAxis->Scale->MajorStep )
                      {
                         zedGraphControl3->GraphPane->XAxis->Scale->Max = time3[n];
                         zedGraphControl3->GraphPane->XAxis->Scale->Min = zedGraphControl3->GraphPane->XAxis->Scale->Max - 5;
                      }
                   }

                }
                zedGraphControl3->AxisChange();
                zedGraphControl3->Invalidate();
                intCounterCanal3++;
                intIncremento3 += (int)flTasaMuestreo;
                if(intIncremento3 >= (flTasaMuestreo * stopTime) )
                {
                   timerGraficar->Enabled = false;
                   HabilitarBotones();
                }
                txtTranscurrido->Text = intCounterCanal1.ToString();
             }
          }
       }
    }

//Función que define el predicado para saber si los elementos del array son cero
static bool IsZero(double dato)
{
```

```
   if(dato != 0)
      return false ;
   else
      return true;
}

/*******************************************************/
/***       Configuración menú Archivo            ***/
/*******************************************************/
private: System::Void salirToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e)
       {
          this->Close();
       }

/*******************************************************/
/***       Configuración menú Adquirir             ***/
/*******************************************************/
private: System::Void informaciónGeneralToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e)
       {

          //Muestra el panel de información como un cuadro de diálogo modal
          //y determina si el resultado es OK
          if(Info->ShowDialog(this) == ::DialogResult::OK)
          {
             //Guarda el nombre del sitio
             varSitio = Info->txtSitio2->Text;
             //Guarda el nombre del operador
             varOperador = Info->txtOperador2->Text;
             //Guarda el nombre del proyecto
             varProyecto = Info->txtProyecto2->Text;
             txtSitio->Text = varSitio;

          }
          //Cierra el cuadro de diálogo
         // delete Info;
       }

/*******************************************************/
/***       Configuración menú Ayuda              ***/
/*******************************************************/
private: System::Void btnAyuda_Click(System::Object^  sender, System::EventArgs^  e)
       {
          Help::ShowHelp(this, helpProvider1->HelpNamespace);
       }
};
}



/*************************************************************************/
/*** ClaseAdquisición.h: header file donde se declaran las funciones  ***/
/***              para la clase Adquisición                ***/
/*************************************************************************/

#pragma once

class Adquisición
{
public:
   Adquisición(void);                  //Constructor
   virtual ~Adquisición(void);            //Destructor
   void CreateTask();                  //Crea el task
   void CreateAIVoltageChan();            //Crea un canal de voltaje análogo
   void CfgSampClkTiming();            //Fija la tasa para el sample clock (modo continuo)
   void CfgInputBuffer();              //Configuración del buffer
   void StartTask();                //Comienza la adquisición
   void ReadAnalog();                  //Lectura de datos
   void IsTaskDone();                  //Averigua si se completó la ejecución del task
   void ClearTask();                //Limpia el task
   void StopTask();                 //Detiene el task
   void Save();                        //Guarda los datos
   void Reset();

   //Parámetros task
   TaskHandle  taskHandle;
   bool32      done;
   time_t      startTime;
   time_t      a;
   char        errBuff[2048];
   int32    error;
   bool32      can;
   //Párametros canal
   char     canales[35];
   // Parametros timing
   uInt64      samplesPerChan;
    float64     sampleRate;
   //Parámetros lectura datos
   //float64     data[100000 * 3];
```

```
   float64     data[size*3];
   float64     datos[size];
   int32       pointsRead;
   int32       totalRead;
    int32       pointsToRead;
   float64     timeout;
private:
   // Parámetros canal
    float64     min;
    float64     max;

};



/*****************************************************************/
/*** ClaseAdquisición.cpp: Archivo de código fuente donde se       ***/
/***  definen las funciones de la clase Adquisición              ***/
/*****************************************************************/

#include "StdAfx.h"
#include "ClaseAdquisición.h"
using namespace System;

Adquisición::Adquisición(void)
{
   //Inicialización de variables
   // Parámetros task
   error = 0;
    taskHandle = 0;
    errBuff[2048]='\0';
    done=0;
   can = false;
   a = time(NULL);
   // Parámetros de canal
   min = -10.0;
    max = 10.0;
   // Parámetros timing
   samplesPerChan = 200;
    sampleRate = 5000;
    // Parámetros lectura datos
   timeout = 10.0;
    totalRead = 0;
   pointsToRead = 5;
}

Adquisición::~Adquisición(void)
{
}

void Adquisición::CreateTask()
{
   DAQmxErrChk (DAQmxBaseCreateTask ("", &taskHandle));

Error:
    if (DAQmxFailed (error))
       DAQmxBaseGetExtendedErrorInfo (errBuff, 2048);
}

void Adquisición::CreateAIVoltageChan()
{
   //char        chan[] = "Dev1/ai0";
   DAQmxErrChk (DAQmxBaseCreateAIVoltageChan (taskHandle, canales, "", DAQmx_Val_Cfg_Default, min, max, DAQmx_Val_Volts, NULL));

Error:
   if (DAQmxFailed (error))
       DAQmxBaseGetExtendedErrorInfo (errBuff, 2048);
}

void Adquisición::CfgSampClkTiming()
{
   char        clockSource[] = "OnboardClock";
   //DAQmxErrChk (DAQmxBaseCfgSampClkTiming(taskHandle,clockSource,sampleRate,DAQmx_Val_Rising,DAQmx_Val_ContSamps,samplesPerChan));
   //DAQmxErrChk (DAQmxBaseCfgSampClkTiming(taskHandle,clockSource,sampleRate,DAQmx_Val_Rising,DAQmx_Val_FiniteSamps,samplesPerChan));
   DAQmxErrChk (DAQmxBaseCfgSampClkTiming(taskHandle,clockSource,sampleRate,DAQmx_Val_Rising,DAQmx_Val_ContSamps,samplesPerChan));
Error:
    if (DAQmxFailed (error))
       DAQmxBaseGetExtendedErrorInfo (errBuff, 2048);
}

void Adquisición::CfgInputBuffer()
{
   DAQmxErrChk (DAQmxBaseCfgInputBuffer(taskHandle,200000)); //use a 100,000 sample DMA buffer

Error:
    if (DAQmxFailed (error))
       DAQmxBaseGetExtendedErrorInfo (errBuff, 2048);
}
```

```
void Adquisición::StartTask()
{
   int32 height = size;
   int32 width = 2;

   DAQmxErrChk (DAQmxBaseStartTask (taskHandle));

   //Inicialización del array data
   for (int n=0;n<height;n++)
   {
      for (int m=0;m<width;m++)
      {
         data[n*width+m] = 0;
      }
   }
   for (int i=0;i<size;i++)
   {
      datos[i] = 0;
    }
Error:
    if (DAQmxFailed (error))
       DAQmxBaseGetExtendedErrorInfo (errBuff, 2048);
}

void Adquisición::ReadAnalog()
{
   //DAQmxErrChk (DAQmxBaseReadAnalogF64 (taskHandle, pointsToRead, timeout, 0, data, size, &pointsRead, NULL));
   ////DAQmxErrChk (DAQmxBaseReadAnalogF64 (taskHandle, pointsToRead, timeout, DAQmx_Val_GroupByScanNumber, data, size, &pointsRead, NULL));
   // totalRead += pointsRead;
   if(can == true)
   {
      DAQmxErrChk (DAQmxBaseReadAnalogF64(taskHandle,pointsToRead,timeout,DAQmx_Val_GroupByScanNumber ,datos,size,&pointsRead,NULL));
      totalRead += pointsRead;
   }
   else
   {
   DAQmxErrChk (DAQmxBaseReadAnalogF64(taskHandle,pointsToRead,timeout,DAQmx_Val_GroupByScanNumber ,data,size * 3,&pointsRead,NULL));
        totalRead += pointsRead;
   }
Error:
    if (DAQmxFailed (error))
       DAQmxBaseGetExtendedErrorInfo (errBuff, 2048);
}

void Adquisición::StopTask()
{
   if(taskHandle != 0)
   {
      DAQmxBaseStopTask (taskHandle);
   }
}

void Adquisición::ClearTask()
{
   if(taskHandle != 0)
   {
      DAQmxBaseClearTask (taskHandle);
   }
}

void Adquisición::IsTaskDone()
{
   DAQmxErrChk(DAQmxBaseIsTaskDone(taskHandle, &done));
Error:
    if (DAQmxFailed (error))
       DAQmxBaseGetExtendedErrorInfo (errBuff, 2048);
}

void Adquisición::Reset()
{
   DAQmxBaseResetDevice ("Dev1");
}
```

ANEXO B. Código interfaz de usuario Análisis y Tratamiento de Acelerogramas (Matlab)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              UNIVERSIDAD DE NARIÑO                           %
%                             FACULTAD DE INGENIERIA                           %
%                           PROGRAMA DE ING. ELECTRONICA                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = Procesamiento1(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Procesamiento1_OpeningFcn, ...
                   'gui_OutputFcn',  @Procesamiento1_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

end

function Procesamiento1_OpeningFcn(hObject, eventdata, handles, varargin)

axes4=imread('udenar.jpg');
axes(handles.axes4);axis off;imshow(axes4);
handles.abrira=0;
handles.gsal=0;

% Choose default command line output for procesamiento1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes procesamiento1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

end
% --- Outputs from this function are returned to the command line.
function varargout = Procesamiento1_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;
end

function bprocesamiento_Callback(hObject, eventdata, handles)
% hObject    handle to bprocesamiento (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    set(handles.uitoolbar1,'Visible','on');
    set(handles.archivo,'Visible','on');
    set(handles.proce,'Visible','on');
    set(handles.grafica,'Visible','on');
    set(handles.ayuda,'Visible','on');
    set(handles.bprocesamiento, 'Visible','off');
    set(handles.badquisicion, 'Visible','off');
    set(handles.bienvenido,'Visible','off');
    set(handles.titulo,'Visible','off');
    set(handles.univer,'Visible','off');
    set(handles.axes5, 'Visible','on');
    set(handles.text6, 'Visible','on');
    set(handles.text7, 'Visible','on');
    set(handles.text4, 'Visible','on');
    set(handles.text5, 'Visible', 'on');
    set(handles.uipanel5, 'Visible','on')
    set(handles.uipanel6,'Visible','on');
    set(handles.uipanel1,'Visible','on');
    set(handles.ayudarap,'String',...
        'Abra un archivo(.txt), proveniente del programa...
         de adquisición y registro');

axes(handles.axes1);cla;set(title('GRÁFICA No.1 - EJE NS'));...
    xlabel('Tiempo (s)'); ylabel('Voltaje (V)'); grid on;
set(handles.axes1, 'Visible', 'on');
axes(handles.axes2);cla;set(title('GRÁFICA No.2 - EJE EW'));...
```

```
    xlabel('Tiempo (s)'); ylabel('Voltaje (V)'); grid on;
set(handles.axes2, 'Visible', 'on');
axes(handles.axes3);cla;set(title('GRÁFICA No.3 - EJE Z'));...
    xlabel('Tiempo (s)'); ylabel('Voltaje (V)'); grid on;
set(handles.axes3, 'Visible', 'on');
axes5=imread('universidad.png');
axes(handles.axes5);axis off;imshow(axes5);

end
% --------------------------------------------------------------------
function abrir_Callback(hObject, eventdata, handles)

[nombre dir]=uigetfile('*.txt', 'Abrir');
%retorna la direccion del archivo a abrir con extensión txt
%retorna 0 si no abrio ningun archivo

if isequal (nombre, 0)  %ejecuta la rutina si se abrio un archivo
    return
else
    axes(handles.axes1);cla;
    axes(handles.axes2);cla;
    axes(handles.axes3);cla;
    nombrearch=strcat(dir,nombre);

    archab=1;
    a =importdata(nombrearch, ' ', 16);
    %importa los datos en textdata para encabezado y en data para No.

    fm=char(a.textdata(15,1));
    %toma la frecuencia de muestreo y la convierte en string
    dur=char(a.textdata(11,1)); % duracion del muestreo para grafica
    ncanales= char(a.textdata(13,1));% numero de canales

    %inicialización de banderas y variables
    corr=0; integ=0;orden=13;auto=0;ej=0;aplii=0;primvez=0;
    handles.cor=corr;handles.ord=orden;handles.inte=integ;
    handles.automatic=auto;handles.eje=ej;handles.apli=aplii;
    handles.prim=primvez;handles.espfrecns=0;handles.espfrecew=0;
    handles.espfrecz=0;handles.hv=0;handles.nsv=0;handles.ewv=0;

    set(handles.text8, 'Visible','on');
    set(handles.text9, 'Visible','on');
    set(handles.text11, 'Visible','on');
    set(handles.text11, 'Visible','on');
    set(handles.text8, 'String', fm);
    set(handles.text9, 'String', dur);
    set(handles.text11, 'String', ncanales);
    set(handles.text10, 'String', nombre);

    % falta sacar el tiempo de adquisicion del encabezado.

    fm= eval(fm); %convierte fm en valor numerico.
    handles.frec=fm; %guarda en handles para posterior uso
    ncanales=eval(ncanales);
    handles.nch=ncanales;
    dur= eval(dur);
    handles.dura=dur;

    ch1= a.data(:,2); %toma los datos del canal 1
    ch2= a.data(:,3); %toma los datos del canal 2
    ch3= a.data(:,4); %toma los datos del canal 3

    %se crea el vector tiempo para realizar la gráfica
    y= length(ch1)
    tiempo = linspace(0,dur,y);
   % tiempo= [0:0.005:(dur-(1/fm))];
    tiempo = double(tiempo);
    tiempo = tiempo';
    t=length(tiempo)

    %se almacena los vectores en funciones handles
    handles.t=tiempo;
    handles.canal1=ch1;
    handles.abrira=archab;
    handles.canal2=ch2;
    handles.canal3=ch3;

    %Se realiza las gráficas
    axes(handles.axes1);plot(tiempo,ch1);...
        set(title('GRÁFICA No.1 - EJE NS'));...
        xlabel('Tiempo (s)'); ylabel('Voltaje (V)'); grid on;
    axes(handles.axes2); plot(tiempo,ch2);...
        set(title('GRÁFICA No.2 - EJE EW'));...
        xlabel('Tiempo (s)'); ylabel('Voltaje (V)'); grid on;
    axes(handles.axes3); plot(tiempo,ch3);...
        set(title('GRÁFICA No.3 - EJE Z'));...
        xlabel('Tiempo (s)'); ylabel('Voltaje (V)'); grid on;
```

```
    %ayuda rapida
    set(handles.ayudarap,'String',...
     'Modo automático: aplica corrección instrumental...
     y corrección de línea de base a los datos');


    %cuadro de diálogo de ajuste automático
    sal = questdlg('¿Desea realizar ajuste en modo automático?',...
        'Corrección de Datos', 'Si', 'No', 'Si'); %mensaje de salida

switch sal
    case 'Si'
        handles.cor=1;
        guidata (hObject, handles); %guarda datos
        instrumental_Callback(hObject, eventdata, handles);
        %llama a corrección instrumental
    case 'No'
        set(handles.ayudarap,'String',...
            'En el menú "procesamiento" encuentra los diferentes...
             procedimientos para el tratamiento de los datos');
        guidata (hObject, handles); %guarda los datos
        return
end
end
end
% --------------------------------------------------------------------
function guardarcalc_Callback(hObject, eventdata, handles)
% hObject    handle to guardaresp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
archab=handles.abrira;

if archab==0
    errordlg('No ha abierto ningún archivo','Error');
else
    efns=handles.espfrecns;
    efew=handles.espfrecew;
    efz=handles.espfrecz;
    cesp1=handles.hv;
    cesp2=handles.nsv;
    cesp3=handles.ewv;

   [nombre, dir]=uiputfile('../*.txt',...
   'Guardar cálculos de espectros de fourier y cocientes espectrales');
    path=strcat(dir,nombre);
    if nombre==0
        return
    else
        archc= fopen(path,'w');
        %se define una matriz con los cálculos de los espectros de fourier
        %y los cocientes espectrales
        b=[efns efew efz cesp1 cesp2 cesp3];

        y=length(efns);
        salida=handles.gsal;

      fprintf(archc,'Universidad de Nariño\n');
      fprintf(archc,'Centro de Investigación en Ingeniería\n');
      fprintf(archc,'Ingeniería Electrónica\n');
      fprintf(archc,'Operador: GIS\n');
      fprintf(archc,'Separador: Doble espacio\n');
      fprintf(archc,'Espectros de frecuencia y cocientes espectrales\n');
      fprintf(archc,'Eje ns  Eje ew  Eje z   H/V   NS/V    EW/V\n');
      fprintf(archc,'%7.7f  %7.7f  %7.7f  %7.7f  %7.7f  %7.7f\n',b');
      fclose(archc)
      guidata(hObject,handles);
        if salida==1
            clear;
            close;
            quit;
        end
    end
end
end
% --------------------------------------------------------------------
function guardarc_Callback(hObject, eventdata, handles)
%cuadro de dialogo guardar como
archab=handles.abrira;

if archab==0
    errordlg('No ha abierto ningún archivo','Error');
else
    [nombre, dir]=uiputfile('../*.txt','Guardar como');
    path=strcat(dir,nombre);
    if nombre==0
        return
    else
```

```
        arch= fopen(path,'w');
        ch1=handles.canal1;
        ch2=handles.canal2;
        ch3=handles.canal3;

        a=[ch1 ch2 ch3];

        y=length(ch1);
        salida=handles.gsal;
        fprintf(arch,'Universidad de Nariño\n');
        fprintf(arch,'Centro de Investigación en Ingeniería\n');
        fprintf(arch,'Ingeniería Electrónica\n');
        fprintf(arch,'Operador: GIS\n');
        fprintf(arch,'Separador: Doble espacio\n');
        fprintf(arch,'Canal 1   Canal2    Canal3\n');
        fprintf(arch,'%7.7f  %7.7f  %7.7f\n',a');
        fclose(arch)
        guidata(hObject,handles);
        if salida==1
            clear;
            close;
            quit;
        end

    end
end
end
% --------------------------------------------------------------------
function salir_Callback(hObject, eventdata, handles)
% permite guardar cambios y salir de la ventana

sal = questdlg('¿Desea guardar los cambios?', 'Salir', 'Si', 'No', 'No');
%mensaje de salida

switch sal
    case 'Si'
        salida=1;
        handles.gsal=salida;
        guidata(hObject,handles);
        guardarc_Callback(hObject, eventdata, handles);

    case 'No'
        clear
        close
        quit
end
end
% --------------------------------------------------------------------
function filtros_Callback(hObject, eventdata, handles)
%muestra el panel de filtros

archab=handles.abrira;

if archab==0
    errordlg('Abra un archivo para aplicar procedimiento',...
        'No hay datos');
    return
elseif archab==1
    set(handles.ayudarap,'String',...
    'Seleccione el tipo de filtro y la correspondiente frecuencia...
     de corte, pulse OK para guardar los cambios');
    set(handles.panelfiltros,'Visible', 'on');
    % muestra el panel de filtros
    set(handles.cpanelfilt, 'Visible','on');
    set(handles.text11, 'Visible', 'off');
    set(handles.filtros,'Checked','on');


elseif archab==2
    warndlg('El panel de filtros ya se encuentra habilitado',...
        'Filtros')
    return
end
end
% --------------------------------------------------------------------
function espfreq_Callback(hObject, eventdata, handles)
% grafica los espectros de frecuencia y los cocientes espectrales

archab=handles.abrira;

if archab==0
    errordlg('Abra un archivo para aplicar procedimiento',...
        'No hay datos');
    return
else
    ch1=handles.canal1;
    ch2=handles.canal2;
    ch3=handles.canal3;
```

```
fm=handles.frec;
dur=handles.dura;

if archab==0
    errordlg('Abra un archivo para aplicar procedimiento',...
        'No hay datos');
    return
else
    dt=1./fm;   %intervalo de tiempo entre muestra y muestra
    %fc=fm/2;
    df=1./dur;

    y= length(ch1);

    freq=(0:y-1)'*df;
    np2=y/2+1;                %mitad de los datos mas uno
    handles.ndat=np2;
    per=freq(2:np2);            %mitad de la frecuencia

    %Componente z
    fz=fft(ch3);
    fzam=abs(fz)*dt; %valor absoluto
    zff=fzam(2:np2);
    handles.espfrecz=zff;

    %Componente ns
    fns=fft(ch1);
    fnsam=abs(fns)*dt;  %valor absoluto
    nsff=fnsam(2:np2);
    handles.espfrecns=nsff;

    %Componente ew
    few=fft(ch2);
    fewam=abs(few)*dt;   %valor absoluto
    ewff=fewam(2:np2);
    handles.espfrecew=ewff;

   %gráficas

    espf=figure(1);
    set(espf,'Name','Espectros de Frecuencia');
    subplot(3,1,3), plot(per,zff,'r-');
    xlabel('Freq (Hz)');
    ylabel('Mod-FFT');
    title('COMPONENTE Z')

    subplot(3,1,1), plot(per,nsff,'r-');
    xlabel('Freq (Hz)');
    ylabel('Mod-FFT');
    title('COMPONENTE NS');

    subplot(3,1,2), plot(per,ewff,'r-');
    xlabel('Freq (Hz)');
    ylabel('Mod-FFT');
    title('COMPONENTE EW');

    set(handles.ayudarap,'String',...
    'Para guardar los cálculos de la FFT, use la opción...
     "guardar cálculos" del menú "archivo"');

    sal = questdlg('¿Desea graficar cocientes espectrales?',...
        'Gráfica', 'Si', 'No', 'No'); %mensaje de salida
    switch sal
        case 'Si'
        %Nakamura
        %Cocientes Espectrales
        hprom=(nsff+ewff)/2;  %promedio componentes horizontales
        nak=hprom./zff;    %promedio horizontal /componente z
        handles.hv=nak;
        nsz=nsff./zff;     %componentens / componentez
        handles.nsv=nsz;
        ewz=ewff./zff;     %componenteew / componentez
        handles.ewv=ewz;

        %proce relaciones espectrales
        cocesp=figure(2);
        set(cocesp,'Name','Cocientes Espectrales');

        subplot(3,2,1), plot(per,nak,'k-');grid on;
        title('Relación espectral H/V');
        xlabel('Frec. (Hz)');
        ylabel('H/V');
        grid on ;

        subplot(3,2,2), loglog(per,nak,'k-');grid on;
        title('Representación logarítmica ');
        xlabel('Frec. (Hz)');
        ylabel('H/V');
```

```
            grid on;

            subplot(3,2,3), plot(per,nsz,'k-');grid on;
            title('Relación espectral NS/Z');
            xlabel('Frec. (Hz)');
            ylabel('NS/V');
            grid on ;

            subplot(3,2,4), loglog(per,nsz,'k-');grid on;
            title('Representación logarítmica ');
            xlabel('Frec. (Hz)');
            ylabel('NS/V');
            grid on;

            subplot(3,2,5), plot(per,ewz,'k-') ;grid on;
            title('Relación espectral EW/Z');
            xlabel('Frec. (Hz)');
            ylabel('EW/V');

            subplot(3,2,6), loglog(per,ewz,'k-');grid on;
            title('Representación logarítmica ');
            xlabel('Frec. (Hz)');
            ylabel('EW/V');

            set(handles.ayudarap,'String',...
            'Para guardar los cálculos de la FFT y los cocientes...
             espectrales, use la opción "guardar cálculos" del menú...
             "archivo"');
         case 'No'
            guidata(hObject,handles);
            return
        end
    end
end
end
% --------------------------------------------------------------------
function ejes_Callback(hObject, eventdata, handles)
% muestra el panel de ejes

archab=handles.abrira;
ej=handles.eje;

if archab==0
    errordlg('Primero abrir archivo','No hay datos');
elseif ej==1
    warndlg('Panel ya ha sido habilitado', 'Panel de ejes');
else
    set(handles.panelejes,'Visible', 'on');
    % muestra el panel de ejes
    set(handles.cpanelejes,'Visible', 'on');
    set(handles.ejes,'Checked', 'on');
    %deshabilitados hasta no quitar el checkbox
    set(handles.radiobutton1, 'Enable', 'off');
    set(handles.radiobutton2, 'Enable', 'off');
    set(handles.radiobutton3, 'Enable', 'off');
    handles.eje=1;
    set(handles.ayudarap,'String',...
    'Desactive la opción todos los ejes para cambiar la visualización');
end
end
% --------------------------------------------------------------------
function proce_Callback(hObject, eventdata, handles)
end
% --------------------------------------------------------------------
function grilla_Callback(hObject, eventdata, handles)
end
% --------------------------------------------------------------------
function checkbox1_Callback(hObject, eventdata, handles)
% para visualizar todos o uno de los ejes en la ventana

a= get(hObject, 'Value'); % a=1 cuando checked muestra todas las graficas
if (a==1)
    ch1=handles.canal1;
    ch2=handles.canal2;
    ch3=handles.canal3;
    tiempo=handles.t;
    set(handles.radiobutton1, 'Enable', 'off');
    set(handles.radiobutton2, 'Enable', 'off');
    set(handles.radiobutton3, 'Enable', 'off');
    set(handles.axes1, 'Visible','on');plot(tiempo,ch1);grid on;
    set(handles.axes2, 'Visible','on');plot(tiempo,ch2);grid on;
    set(handles.axes3, 'Visible','on');plot(tiempo,ch3);grid on;
else %a=0 cuando unchecked, muestra la grafica del eje NS
    set(handles.radiobutton1, 'Enable', 'on');
    set(handles.radiobutton2, 'Enable', 'on');
    set(handles.radiobutton3, 'Enable', 'on');
    set(handles.radiobutton1, 'Value',1);
    set(handles.radiobutton2, 'Value',0);
```

```
    set(handles.radiobutton3, 'Value',0);
    set(handles.axes2, 'Visible','off');
    axes(handles.axes2);cla;
    set(handles.axes3, 'Visible','off');
    axes(handles.axes3);cla;
end
end
%-----------------------------------------------------------
function ok_Callback(hObject, eventdata, handles)
%Boton OK del panel filtros
guidata(hObject,handles);

set(handles.panelfiltros,'Visible', 'off'); % muestra el panel de filtros
set(handles.cpanelfilt,'Visible', 'off');
set(handles.text11, 'Visible', 'off');
set(handles.filtros,'Checked','off');
ch1=handles.canal1f;
ch2=handles.canal2f;
ch3=handles.canal3f;
tiempo=handles.t;

axes(handles.axes1); plot(tiempo,ch1,'b','LineWidth',1);...
    set(title('GRAFICA No.1 - EJE NS'));xlabel('tiempo (s)');...
    ylabel('aceleracion (mg)'); grid on;
axes(handles.axes2); plot(tiempo,ch2,'b','LineWidth',1);...
    set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');...
    ylabel('aceleracion (mg)'); grid on;
axes(handles.axes3); plot(tiempo,ch3,'b','LineWidth',1);...
    set(title('GRAFICA No.3 - EJE Z')); xlabel('tiempo (s)');...
    ylabel('aceleracion (mg)'); grid on;
set(handles.text11, 'Visible', 'off');

handles.canal1=ch1;
handles.canal2=ch2;
handles.canal3=ch3;

guidata(hObject,handles);
end
%-----------------------------------------------------------
function cancel_Callback(hObject, eventdata, handles)
%Boton cancelar del panel filtros

ch1=handles.canal1;
ch2=handles.canal2;
ch3=handles.canal3;
tiempo=handles.t;

set(handles.panelfiltros, 'Visible', 'off');
set(handles.text11, 'Visible', 'off');
set(handles.filtros,'checked','off');
axes(handles.axes1); plot(tiempo,ch1); ...
    set(title('GRAFICA No.1 - EJE NS'));xlabel('tiempo (s)'); ...
    ylabel('aceleracion (mg)'); grid on;
axes(handles.axes2); plot(tiempo,ch2);...
    set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');...
    ylabel('aceleracion (mg)'); grid on;
axes(handles.axes3); plot(tiempo,ch3);...
    set(title('GRAFICA No.3 - EJE Z'));xlabel('tiempo (s)');...
    ylabel('aceleracion (mg)'); grid on;
end
% --------------------------------------------------------------------
function radiobutton1_Callback(hObject, eventdata, handles)
%para visualización en pantalla únicamente del eje NS

rb1=get(hObject, 'Value'); % toma el estado del eje x, 1 si esta activado.

if (rb1 == 1)

    tiempo=handles.t;
    ch1=handles.canal1;

    set(handles.axes1, 'Visible','on');plot(tiempo, ch1);...
        set(title('GRAFICA No.1 - EJE NS'));xlabel('tiempo (s)');...
        ylabel('aceleracion (mg)'); grid on;
    axes(handles.axes2);cla;
    axes(handles.axes3);cla;
    set(handles.axes2, 'Visible','off');
    set(handles.axes3, 'Visible','off');

else
    return
end
end
% --------------------------------------------------------------------
function radiobutton2_Callback(hObject, eventdata, handles)
%para visualización en pantalla únicamente del eje EW

rb2=get(hObject,'Value');
```

```
if (rb2 == 1) % toma el estado del eje NS, 1 si esta activado.

    tiempo=handles.t;
    ch2=handles.canal2;

    set(handles.axes3);cla;
    set(handles.axes1);cla;
    set(handles.axes3, 'Visible','off');
    set(handles.axes1, 'Visible','off');


    set(handles.axes2, 'Visible','on');...
    set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');...
     ylabel('aceleracion (mg)');
    axes(handles.axes2); plot(tiempo,ch2);grid on;

else
    return
end
end
% --------------------------------------------------------------------
function radiobutton3_Callback(hObject, eventdata, handles)
%para visualización en pantalla únicamente del eje Z

rb3=get(hObject, 'Value'); % toma el estado del eje EW, 1 si esta activado.

if (rb3 == 1)

    tiempo=handles.t;
    ch3=handles.canal3;

    set(handles.axes3, 'Visible','on');...
    set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');...
    ylabel('aceleracion (mg)');
    axes(handles.axes3); plot(tiempo,ch3);grid on;
    set(handles.axes1, 'Visible','off');
    axes(handles.axes1);cla;
    set(handles.axes2, 'Visible','off');cla;
    axes(handles.axes2);cla
else
    return
end
end

% --- Executes on mouse press over axes background.
function axes1_ButtonDownFcn(hObject, eventdata, handles)

set(handles.axes1, 'Selected','on');
set(handles.axes2, 'Selected', 'off');
set(handles.axes3, 'Selected', 'off');
end
%-----------------------------------------------------
function axes2_ButtonDownFcn(hObject, eventdata, handles)

set(handles.axes2, 'Selected','on');
set(handles.axes1, 'Selected', 'off');
set(handles.axes3, 'Selected', 'off');
end
% --- Executes on mouse press over axes background.
function axes3_ButtonDownFcn(hObject, eventdata, handles)

set(handles.axes1, 'Selected', 'off');
set(handles.axes2, 'Selected', 'off');
set(handles.axes3, 'Selected','on');
end

% --------------------------------------------------------------------
function actgrid_Callback(hObject, eventdata, handles)

    axes(handles.axes1); grid on;
    axes(handles.axes2); grid on;
    axes(handles.axes3); grid on;
end
% --------------------------------------------------------------------
function desactgrid_Callback(hObject, eventdata, handles)

    axes(handles.axes1); grid off;
    axes(handles.axes2); grid off;
    axes(handles.axes3); grid off;
end
% --------------------------------------------------------------------
function zoom_Callback(hObject, eventdata, handles)
end
% --------------------------------------------------------------------
function zoomin_Callback(hObject, eventdata, handles)

pan off
```

```
zoom xon
end
% --------------------------------------------------------------------
function zoomout_Callback(hObject, eventdata, handles)

pan off
zoom xout
end
% --------------------------------------------------------------------
function desactivar_Callback(hObject, eventdata, handles)

zoom off
end

% --------------------------------------------------------------------
function opciones_Callback(hObject, eventdata, handles)
end
% --------------------------------------------------------------------
function ayuda_Callback(hObject, eventdata, handles)
%Llama al archivo común de ayuda de extensión *.chm
!Ayuda.chm
end
% --------------------------------------------------------------------
function correccion_Callback(hObject, eventdata, handles)
% corrección de linea de base desde menú

archab=handles.abrira;
integ=handles.inte;
aplii=handles.apli;
dur=handles.dura;
primvez=handles.prim;

if archab==0
    errordlg('Abra un archivo para aplicar procedimiento','Error');
    return
elseif archab==1

    ch1=handles.canal1;
    ch2=handles.canal2;
    ch3=handles.canal3;
    tiempo=handles.t;

    set(handles.ayudarap,'String','Espere un momento por favor...
    Realizando corrección de línea base');

    %se elimina el offset y las tendencias
   ch1=detrend(ch1);
   ch2=detrend(ch2);
   ch3=detrend(ch3);

    if primvez==0 %si es la primera vez que se aplica el procedimiento...
        %conversión de voltaje a aceleración.
        ch1 = (ch1/(2.7953*0.498))*1000; %para x
        ch2 = (ch2/(2.8371*0.499))*1000; %para y
        ch3 = (ch3/(2.7992*0.5))*1000;   %para z
        primvez=1;
        handles.prim=primvez;
    end

    k=length(ch1);
        if integ<=1
            for i=1:k
                ch1n(i+1)=ch1(i);
                ch2n(i+1)=ch2(i);
                ch3n(i+1)=ch3(i);
            end
   %cumple con condición de borde inicial
        ch1n(1)=0;
        ch2n(1)=0;
        ch3n(1)=0;
   %cumple con condición de borde final
        ch1n(k+2)=0;
        ch2n(k+2)=0;5
        ch3n(k+2)=0;

        ch1=ch1n';
        ch2=ch2n';
        ch3=ch3n';

        sep=tiempo(2)-tiempo(1);
        tiempo(k+1)=(tiempo(k)+sep);
        tiempo(k+2)=(tiempo(k+1)+sep);
    end

    axes(handles.axes1); plot(tiempo, ch1); xlim([0,dur]);
    set(title('GRAFICA No.1 - EJE NS'));xlabel('tiempo (s)');...
    ylabel('aceleracion (mg)');grid on;
    axes(handles.axes2); plot(tiempo, ch2); xlim([0,dur]);
```

```
    set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');...
    ylabel('aceleracion (mg)');grid on;
    axes(handles.axes3); plot(tiempo, ch3); xlim([0,dur]);
    set(title('GRAFICA No.3 - EJE Z'));xlabel('tiempo (s)');...
    ylabel('aceleracion (mg)');grid on;
    set(handles.ayudarap,'String','En el menú "procesamiento" puede...
    encontrar diferentes procedimientos para el tratamiento de los datos');

    if aplii==1
        corr=1;
        handles.cor=corr;
        handles.canal1=ch1;
        handles.canal2=ch2;
        handles.canal3=ch3;
        handles.apli=1;
        handles.t=tiempo;
        guidata(hObject,handles);
        instrumental_Callback(hObject, eventdata, handles);
    else
        corr=1;
        handles.cor=corr;
        handles.canal1=ch1;
        handles.canal2=ch2;
        handles.canal3=ch3;
        handles.t=tiempo;
        guidata(hObject,handles);
    end
end
end
% --------------------------------------------------------------------
function integ_Callback(hObject, eventdata, handles)
%rutina para la integración con el fin de obtener diagramas de velocidades
%y desplazamientos.

archab=handles.abrira;
dur=handles.dura;
if archab==0
    errordlg('Abra un archivo para aplicar procedimiento','Error');
else
    fm=handles.frec;
    ch1=handles.canal1;
    ch2=handles.canal2;
    ch3=handles.canal3;
    integ=handles.inte;
    corr=handles.cor; %si se ha realizado correccion de linea de base?


    if corr==0
        errordlg('Corrección de linea de base no aplicada', 'Error');
    elseif integ<2
        if integ==0
            %Conversión de datos en cm/s^2
            ch1=ch1*0.980655;
            ch2=ch2*0.980655;
            ch3=ch3*0.980655;
        end
        ns=length(ch1);
        t1=(0:(1/fm):((ns/fm)-(1/fm)))';
        length(t1)

        set(handles.ayudarap,'String','Espere un momento por favor...
        realizando proceso de integración');
        integns(1)=0;
        integew(1)=0;
        integz(1)=0;

        for i=1:(ns-1)
                calcns(i)=((ch1(i)+ch1(i+1))/2)*(1/(fm));
                calcew(i)=((ch2(i)+ch2(i+1))/2)*(1/(fm));
                calcz(i)=((ch3(i)+ch3(i+1))/2)*(1/(fm));

                integns(i+1)= calcns(i)+integns(i);
                integew(i+1)= calcew(i)+integew(i);
                integz(i+1)= calcz(i)+integz(i);
        end

        if integ==0
            axes(handles.axes1); plot(t1, integns); xlim([0 dur]);
            set(title('GRAFICA No.1 - EJE NS'));xlabel('tiempo (s)');...
             ylabel('velocidad (cm/s)');grid on;
            axes(handles.axes2); plot(t1, integew);xlim([0,dur]);
            set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');...
            ylabel('velocidad (cm/s)');grid on;
            axes(handles.axes3); plot(t1, integz);xlim([0,dur]);
            set(title('GRAFICA No.3 - EJE Z'));xlabel('tiempo (s)');...
            ylabel('velocidad (cm/s)');grid on;
            integ=1;
            set(handles.ayudarap,'String','Si desea puede integrar...
```

```
            nuevamente los datos para obtener diagramas de desplazamiento')
        elseif integ==1
            axes(handles.axes1); plot(t1, integns);
            set(title('GRAFICA No.1 - EJE NS'));xlabel('tiempo (s)');...
            label('desplazamiento (cm)');grid on;
            axes(handles.axes2); plot(t1, integew);
            set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');...
            ylabel('desplazamiento (cm)');grid on;
            axes(handles.axes3); plot(t1, integz);
            set(title('GRAFICA No.3 - EJE Z'));xlabel('tiempo (s)');...
            ylabel('desplazamiento (cm)');grid on;
            integ=2;
            set(handles.ayudarap,'String','En el menú "procesamiento"...
            puede encontrar diferentes procedimientos para el tratamiento..
            de los datos');

        end

        handles.cor=0;
        handles.canal1=integns';
        handles.canal2=integew';
        handles.canal3=integz';
        handles.inte=integ;
        handles.t=t1;
        guidata(hObject,handles);

    else
        errordlg('No es posible realizar mas procesos de integración')
    end
end
end
% --------------------------------------------------------------------
function ordfilt_Callback(hObject, eventdata, handles)
%ventana emergente para cambio de orden del filtro

    set(handles.ayudarap,'String','El orden calculado para un...
    funcionamiento óptimo de filtro es 16. se recomienda no cambiar este...
    valor si no es necesario');
    prompt={'Ingrese un valor entre 10 y 20:'};
    name='Cambiar orden de filtro';
    numlines=1;
    defaultanswer={'0'};
    answer=inputdlg(prompt,name,numlines,defaultanswer);

    orden = answer{1};

    % proteccion contra ingreso de caracteres
    r=isletter(orden);
    n=length(r);
    letra = 0;
    for i = 1:n, letra = letra + r(i); end ;

    if letra>=1
        errordlg('Ingrese un valor numérico');
        pause
        ordfilt_Callback();
    else
        orden= str2num(orden);

        if orden==0
            errordlg('Ingrese un valor diferente de cero', 'Error');
            pause
            ordfilt_Callback();
            return

        elseif orden>=20
            errordlg('Ingrese un valor menor a 20', 'Error');
            pause
            ordfilt_Callback();
            return

        elseif orden <10
            errordlg('Ingrese un valor mayor o igual a 10', 'Error');
            pause
            ordfilt_Callback();
            return
        else
            handles.ord=orden;
            guidata(hObject,handles);
        end

    end
end
% --------------------------------------------------------------------
function selfilt_Callback(hObject, eventdata, handles)
%pop-up menu para seleccionar el tipo de filtro

ch1=handles.canal1;
```

```
ch2=handles.canal2;
ch3=handles.canal3;
orden=handles.ord;
tiempo=handles.t;

filt=get(handles.selfilt,'value');

switch filt
    case 1
        return
    case 2
        set(handles.ayudarap,'String',...
            'F.pasa altas: permite el paso de frecuencias por encima de...
             la frecuencia de corte que usted elija. Para guardar cambios..
             presione OK');
        prompt={'Ingrese el valor de la frecuencia de corte(Hz):'};
        name='Filtro pasa altas';
        numlines=1;
        defaultanswer={'0'};
        answer=inputdlg(prompt,name,numlines,defaultanswer);

        valfilt = answer{1};

        % proteccion contra ingreso de caracteres
        r=isletter(valfilt);
        n=length(r);
        letra = 0;
        for i = 1:n, letra = letra + r(i); end ;

        if letra>=1
            errordlg('Ingrese un valor numérico');
            return
        else
            valfilt= str2num(valfilt);

        if valfilt==0
            errordlg('Ingrese un valor diferente de cero', 'Error');
                    return
        elseif valfilt>=100
            errordlg('Ingrese un valor menor a 100 Hz', 'Error');
            return
        else
            Wn= valfilt/100; %normalizar frecuencia
            [b,a]=butter(orden,Wn,'high'); %calcula coeficientes del filtro
            ch1=filtfilt(b,a,ch1); %aplica el filtro al canal 1
            ch2=filtfilt(b,a,ch2); %aplica el filtro al canal 2
            ch3=filtfilt(b,a,ch3); %aplica el filtro al canal 3
            archab=2;
            %graficar resultados
            axes(handles.axes1); plot(tiempo,ch1,'b','LineWidth',1);...
            set(title('GRAFICA No.1 - EJE NS'));xlabel('tiempo (s)');...
            ylabel('aceleracion (mg)'); grid on;
            axes(handles.axes2); plot(tiempo,ch2,'b','LineWidth',1);...
            set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');...
            ylabel('aceleracion (mg)'); grid on;
            axes(handles.axes3); plot(tiempo,ch3,'b','LineWidth',1);...
            set(title('GRAFICA No.3 - EJE Z'));xlabel('tiempo (s)');...
            ylabel('aceleracion (mg)'); grid on;
            %se visualiza label filtro activado
            set(handles.text11, 'Visible', 'on');

            handles.canal1f=ch1;
            handles.canal2f=ch2;
            handles.canal3f=ch3;
            guidata(hObject,handles);
        end
        end

    case 3
        set(handles.ayudarap,'String',...
        'F. Pasa bajas: permite el paso de frecuencias por debajo de la...
         frecuencia de corte que usted elija. Para guardar cambios...
         presione OK');
        prompt={'Ingrese el valor de la frecuencia de corte (Hz):'};
        name='Filtro pasa bajas';
        numlines=1;
        defaultanswer={'0'};
        answer=inputdlg(prompt,name,numlines,defaultanswer);

       valfilt = answer{1};

        % proteccion contra ingreso de caracteres
        r=isletter(valfilt);
        n=length(r);
        letra = 0;
        for i = 1:n, letra = letra + r(i); end ;

         if letra>=1
```

```
        errordlg('Ingrese un valor numérico');
        return
     else
        valfilt= str2num(valfilt);

        if valfilt==0
            errordlg('Ingrese un valor diferente de cero', 'Error');
            set(handles.pbajas,'Value',0)
            return
        elseif valfilt>=100
            errordlg('Ingrese un valor menor a 100 Hz', 'Error');
            return
        else
        Wn= valfilt/100; %normalizar frecuencia
        [b,a]=butter(orden,Wn,'low'); %calcula coeficientes del filtro
        ch1f=filtfilt(b,a,ch1); %aplica el filtro al canal 1
        ch2f=filtfilt(b,a,ch2); %aplica el filtro al canal 2
        ch3f=filtfilt(b,a,ch3); %aplica el filtro al canal 3
        archab=2;

        %graficar resultados
        axes(handles.axes1); plot(tiempo,ch1f,'b','LineWidth',1);...
        set(title('GRAFICA No.1 - EJE NS'));xlabel('tiempo (s)');...
         ylabel('aceleracion (mg)'); grid on;
        axes(handles.axes2); plot(tiempo,ch2f,'b','LineWidth',1);...
        set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');...
         ylabel('aceleracion (mg)'); grid on;
        axes(handles.axes3); plot(tiempo,ch3f,'b','LineWidth',1);...
        set(title('GRAFICA No.3 - EJE Z'));xlabel('tiempo (s)');...
         ylabel('aceleracion (mg)'); grid on;

        %se visualiza label filtro activado
        set(handles.text11, 'Visible', 'on');

        handles.canal1f=ch1f;
        handles.canal2f=ch2f;
        handles.canal3f=ch3f;
        guidata(hObject,handles);

        end
     end

case 4
    set(handles.ayudarap,'String',...
    'F.Pasa banda: permite el paso de frecuencias entre la frecuencia..
     de corte inferior y superior que usted elija en este momento');
    prompt={'Ingrese frecuencia de corte inferior(Hz)...
    mayor a 0 (Hz)','Ingrese frecuencia de corte superior (Hz)...
    menor a 100 Hz'};
    name='Filtro pasa banda';
    numlines=1;
    defaultanswer={'0','0'};
    answer=inputdlg(prompt,name,numlines,defaultanswer);

    valfilt1= answer{1}; %guarda el valor inferior
    valfilt2= answer{2}; %guarda el valor superior

    % proteccion contra ingreso de caracteres
    letra = strcat (valfilt1,valfilt2);
    r=isletter(letra); % string con 1 en cada lugar donde hay una letra
    n=length(r); % valor de la longitud de r
    letra = 0;
    for i = 1:n, letra = letra + r(i) ; end ; %recorre r para...
    % encontrar el numero letras

    if letra>=1
        errordlg('Ingrese un valor numérico');
        return
    else
        valfilt1= str2num(valfilt1);
        valfilt2= str2num(valfilt2);

    if valfilt1==0
            errordlg('Ingrese un valor diferente de cero', 'Error');
            set(handles.pband,'Value',0)
            return
        elseif valfilt2==0
            errordlg('Ingrese un valor diferente de cero', 'Error');
            set(handles.pband,'Value',0)
            return
        elseif valfilt1>=valfilt2
            errordlg('El limite inferior no debe ser mayor o igual...
             al limite superior', 'Error')
            return
        elseif valfilt1>=100
            errordlg('Ingrese un valor menor de 100 Hz', 'Error');
            return
        elseif valfilt2>=100
```

```
                errordlg('Ingrese un valor menor de 100 Hz', 'Error');
                return
            else
                Wn= [valfilt1/100 valfilt2/100]; %normalizar frecuencia
                [b,a]=butter(orden,Wn); %calcula coeficientes del filtro
                ch1f=filtfilt(b,a,ch1); %aplica el filtro al canal 1
                ch2f=filtfilt(b,a,ch2); %aplica el filtro al canal 2
                ch3f=filtfilt(b,a,ch3); %aplica el filtro al canal 3
                archab=2;

                %graficar resultados
                axes(handles.axes1); plot(tiempo,ch1f,'b','LineWidth',1);..
                set(title('GRAFICA No.1 - EJE NS'));xlabel('tiempo (s)');..
                ylabel('aceleracion (mg)'); grid on;
                axes(handles.axes2); plot(tiempo,ch2f,'b','LineWidth',1);..
                set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');..
                ylabel('aceleracion (mg)'); grid on;
                axes(handles.axes3); plot(tiempo,ch3f,'b','LineWidth',1);..
                set(title('GRAFICA No.3 - EJE Z'));xlabel('tiempo (s)');...
                ylabel('aceleracion (mg)'); grid on;

                %se visualiza label filtro activado
                set(handles.text11, 'Visible', 'on');

                handles.canal1f=ch1f;
                handles.canal2f=ch2f;
                handles.canal3f=ch3f;
                guidata(hObject,handles);
        end
        end
end
end
% --------------------------------------------------------------------
function selfilt_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
         get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

end
% --------------------------------------------------------------------
function automatic_Callback(hObject, eventdata, handles)
%ajuste automatico de datos desde el menú

instru= handles.instrumental;

if instru==1
    warndlg('El procedimiento ya ha sido aplicado');
else
    handles.cor=1;
    guidata(hObject,handles);
    instrumental_Callback(hObject, eventdata, handles);
end
end
% --------------------------------------------------------------------
function instrumental_Callback(hObject, eventdata, handles)
%corrección instrumental con la funcion de transferencia del equipo

archab=handles.abrira;
corr=handles.cor;
aplii=handles.apli;

if archab==0
    errordlg('Abrir archivo para aplicar procedimiento', 'No hay datos');
else
    instru=handles.instrumental;

    if instru==1
        errordlg('El procedimiento ya ha sido aplicado')
    else
        if corr==0
            handles.apli=1;
            guidata(hObject,handles);
            correccion_Callback(hObject,eventdata,handles);
        end
        ch1=handles.canal1;
        ch2=handles.canal2;
        ch3=handles.canal3;
        tiempo=handles.t;

        [B,A] = butter(8,0.2,'low');

        ch1 = filter(B,A,ch1);
        ch2 = filter(B,A,ch2);
        ch3 = filter(B,A,ch3);

        %noise variance
```

```
        nvz = 8.8795e-008;
        nvy = 1.5085e-007;
        nvx = 8.1182e-008;

        %ruido blanco gaussiano con noise variance
        noisez = random('norm', 0, sqrt(nvz),size(ch3),1);
        noisey = random('norm', 0, sqrt(nvy),size(ch2),1);
        noisex = random('norm', 0, sqrt(nvx),size(ch1),1);

        %Coeficientes funciones de transferencia del modelo
           %dinámico para cada eje
        numz = [0  7.437e-005 -0.0002595 0.0004219  -0.0003373 0.0001141];
        denz = [1  -3.988  6.866  -6.321  3.108  -0.657];
        numy = [0  4.059e-005 -0.0001517  0.0002575 -0.0002049  6.348e-005]
        deny = [1 -4.094  7.156  -6.626  3.236  -0.6655];
        numx = [0  4.319e-005 -0.0002146 0.0004067  -0.0003589  0.0001289];
        denx = [1  -4.004  6.87  -6.264   3.024  -0.62];

        %Coeficientes de la función de transferencia de ruido para cada eje
        nnz = [1  0.4471  0.1511 0.3576 -0.1748 0.1716];
        dnz = denz;
        nny = [1  0.1897  -0.03695  0.3916  -0.2148 0.1605];
        dny = deny;
        nnx = [1  0.3467 0.07582 0.3652  -0.1712 0.1926];
        dnx = denx;

        %aplicamos filtro a cada eje teniendo en cuenta el ruido
        ch1 = filter(numx,denx,ch1) + filter(nnx,dnx,noisex);
        ch2 = filter(numy,deny,ch2) + filter(nny,dny,noisey);
        ch3 = filter(numz,denz,ch3) + filter(nnz,dnz,noisez);
        instru=1;

        handles.instrumental=instru;
        handles.canal1=ch1;
        handles.canal2=ch2;
        handles.canal3=ch3;
        corr=1;
        handles.cor=corr;
        axes(handles.axes1); plot(tiempo, ch1);...
           set(title('GRAFICA No.1 - EJE NS'));xlabel('tiempo (s)');...
             ylabel('aceleracion (mg)');grid on;
        axes(handles.axes2); plot(tiempo, ch2);...
           set(title('GRAFICA No.2 - EJE EW'));xlabel('tiempo (s)');...
             ylabel('aceleracion (mg)');grid on;
        axes(handles.axes3); plot(tiempo, ch3);
           set(title('GRAFICA No.3 - EJE Z'));xlabel('tiempo (s)');...
             ylabel('aceleracion (mg)');grid on;
        guidata(hObject, handles);

        set(handles.ayudarap,'String','Si lo desea puede aplicar otro ...
            procedimiento, escoja en el menú "procesamiento"');
    end
end
end
% --------------------------------------------------------------------
function badquisicion_Callback(hObject, eventdata, handles)
%abre el programa de adquisición y registro
!InterfazAdquisición
end
% --------------------------------------------------------------------
function cerrarejes_Callback(hObject, eventdata, handles)
set(handles.panelejes, 'Visible', 'off');
end
% --------------------------------------------------------------------
function figure1_CloseRequestFcn(hObject, eventdata, handles)
delete(hObject);
end
% --------------------------------------------------------------------
function archivo_Callback(hObject, eventdata, handles)
end
% --------------------------------------------------------------------
function grafica_Callback(hObject, eventdata, handles)
end
% --------------------------------------------------------------------
function activar_Callback(hObject, eventdata, handles)

axes(handles.axes1);grid on;
axes(handles.axes2);grid on;
axes(handles.axes3);grid on;
end
% --------------------------------------------------------------------
function cpanelejes_Callback(hObject, eventdata, handles)

set(handles.panelejes, 'Visible', 'off');
set(handles.ejes, 'checked', 'off');
handles.eje=0; %panel no visible
guidata(hObject,handles);
end
```

```
% --------------------------------------------------------------------
function cpanelfilt_Callback(hObject, eventdata, handles)

set(handles.panelfiltros,'Visible','off');
end
% --------------------------------------------------------------------
function uipushtool1_ClickedCallback(hObject, eventdata, handles)

abrir_Callback(hObject, eventdata, handles);
end
% --------------------------------------------------------------------
function uipushtool2_ClickedCallback(hObject, eventdata, handles)

guardarc_Callback(hObject, eventdata, handles)
end
```