UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA INDUSTRIAL
ESPECIALIDAD AUTOMÁTICA Y ELECTRÓNICA INDUSTRIAL

PROYECTO FINAL DE CARRERA

HERRAMIENTA PARA LA ADQUISICIÓN, PROCESAMIENTO Y MONITORIZACIÓN DE SEÑALES. DETECCIÓN DE FALLOS EN EJES Y RODAMIENTOS MECÁNICOS

AUTOR: JAVIER ROMERO CARRASCO
TUTOR: RAMÓN IGNACIO BARBER CASTAÑO
COTUTORA: MARÍA JESÚS GÓMEZ GARCÍA

Octubre de 2011

Son muchas las manos y los corazones
que contribuyen al éxito de una persona.

Walt Disney

El futuro pertenece a las personas
que creen en la belleza de sus sueños.

Eleanor Roosevelt

Agradecimientos

Quiero dar las gracias a mis padres, por mi oportunidad de existir, por su sacrificio en algún tiempo incomprendido, por su ejemplo de superación incansable, por su comprensión y confianza, por su amor y amistad incondicional, porque sin su apoyo no hubiera sido posible la culminación de mi carrera profesional. Por lo que ha sido y será… Gracias

Expresar mi más profundo agradecimiento a mi familia, porque gracias a su cariño, guía y apoyo he llegado a realizar uno de los anhelos más grandes de la vida, fruto del inmerso apoyo, amor y confianza que en mi se depositó y con los cuales he logrado terminar mis estudios profesionales que constituyen el legado más grande que pudiera recibir y por lo cual les viviré eternamente agradecido. Con cariño y respeto.

Gracias Nazareth, eres de esa clase de personas que todo lo comprenden y dan lo mejor de sí mismos sin esperar nada a cambio, porque sabes escuchar y brindar ayuda cuando es necesario, por estar siempre ahí de forma incondicional, por todo esto y mucho más, te quiero.

A todos mis amigos de Madrid, Badajoz y de Murcia, por su preocupación e interés, y por los grandes momentos que hemos vivido y disfrutado juntos.

A todos mis profesores y compañeros de la Universidad, que han hecho mucho más ameno el camino de mis estudios, por su amistad, compañerismo y respeto.

En especial, quiero dar las gracias a Alberto Navarro, mi compañero de clase, por dar siempre lo mejor de sí mismo, ofrecer su ayuda incondicional y porque todos los momentos que he compartido con él, me han hecho crecer como persona.

Principalmente, dar especial agradecimiento a mi tutor Ramón Barber, por la oportunidad que me ha brindado para realizar este proyecto y por la confianza que ha depositado en mí. Del mismo modo, agradecer todo el tiempo invertido en mí, por su paciencia e interés y por su apoyo moral en los momentos más difíciles.

Por último, agradecer a María Jesús Gómez todo el tiempo y esfuerzo que ha dedicado para poder llevar a cabo las prácticas y pruebas del presente proyecto y por ser tan amable y cordial en todos los momentos que hemos compartido en el laboratorio de prácticas.

A todos ellos, mi más sincero agradecimiento. Gracias.

Resumen

El presente Proyecto de Fin de Carrera, se centra en el desarrollo de una interfaz gráfica, la cual permite integrar en una única herramienta gran parte de las aplicaciones necesarias para poder llevar a cabo un estudio pormenorizado aplicado al mantenimiento predictivo de las vibraciones mecánicas que se generan en los rodamientos, pudiendo con dicha herramienta detectar el tiempo de estabilización de éstos y trabajar con vista a corregirlas, aumentando de este modo el confort y la durabilidad de los mismos.

La interfaz gráfica se desarrolla según las especificaciones indicadas por los técnicos y profesores que van a utilizar la herramienta, lo que supone el diseño y desarrollo de un software muy específico que cubra perfectamente la finalidad de sus funcionalidades. Esta interfaz se ejecuta en un PC y se ha desarrollado bajo entorno de programación MATLAB.

La herramienta dispone de varios modos de funcionamiento:

1. La tarjeta de adquisición de datos "KEITHLEY". Se conecta mediante USB al PC permitiendo la captura de datos tanto analógicos como digitales en tiempo real con un rango de 1 a 8 canales. De esta forma, se procesan, modifican y monitorizan los datos reales adquiridos.
2. La tarjeta de sonido del micrófono integrada en el PC. Permite la captura de datos mediante dos canales independientes de cualquier dispositivo que se conecte a la clavija propia del micrófono.
3. Modo simulación.

El proyecto forma parte de los trabajos desarrollados en conjunto por el Departamento de Ingeniería Mecánica y el Departamento de Automática y Sistemas de la Universidad Carlos III de Madrid.

Índice General

Índice de Figuras

1. Introducción

Este Proyecto de Fin de Carrera forma parte de los trabajos desarrollados en conjunto por el Departamento de Ingeniería Mecánica y el Departamento de Automática y Sistemas de la Universidad Carlos III de Madrid.

Dentro de este proyecto, se encuentra la tarea de "Desarrollo de nuevas técnicas para la detección de fallos en los rodamientos" [14], la cual ha sido investigada y desarrollada por la Universidad Carlos III de Madrid para la corrección de las vibraciones mecánicas y mejora de la duración de los rodamientos mecánicos.

En este capítulo se introduce al lector en el marco del estudio aplicado a la adquisición de datos analógicos para mejorar el mantenimiento predictivo de las vibraciones mecánicas que se generan en los rodamientos, dando una visión global de las razones por las cuales surge esta investigación y de los objetivos que persigue. A continuación, se expondrán los objetivos y las motivaciones que persigue el presente Proyecto de Fin de Carrera.

1.1 Motivaciones

Los actuales métodos que se emplean para calcular los tiempos de establecimiento de los ejes y rodamientos y la captación de las vibraciones que producen éstos no son en tiempo real lo que dificulta el proceso de adquisición, comprensión y aprovechamiento de los datos obtenidos de los rodamientos mecánicos.

Además, no se dispone de ninguna herramienta capaz de integrar todas las funcionalidades necesarias para la captación y el tratamiento de éstas señales de manera intuitiva y productiva.

Por tanto, es un hecho comprobado que los sistemas de adquisición de datos para rodamientos y ejes no están optimizados, por ello, surge la necesidad de desarrollar un nuevo sistema que permita el seguimiento y captación de señales en tiempo real con un margen de error mínimo y una presentación de datos fácil, compresible y amena para el usuario.

Todos estos inconvenientes mencionados anteriormente han repercutido en la creación del proyecto "Herramienta para la adquisición, procesamiento y monitorización de señales. Detección de fallos en ejes y rodamientos mecánicos" para que se encargue de mejorar la precisión y la eficacia de los sistemas actuales. Además este proyecto debe hacerse cargo del desarrollo de un software lo suficientemente sofisticado para poder recrear en una interfaz gráfica situaciones prácticas y simuladas lo más realistas posibles.

El presente Proyecto de Fin de Carrera, "Herramienta para la adquisición, procesamiento y monitorización de señales. Detección de fallos en rodamientos mecánicos", es una parte muy importante para el proceso de captura de datos de elementos mecánicos en el Departamento de Ingeniería Mecánica de la Universidad Carlos III de Madrid. Por tanto, esta interfaz gráfica externa tiene que ser lo más explícita y compresible posible para que el usuario se adecue lo más rápido posible a ella y pueda aumentar el rendimiento de los procesos.

Si los resultados obtenidos están acordes con los resultados previstos, se espera un salto tecnológico y revolucionario en los trabajos con ejes y rodamientos mecánicos y el control de sus posibles vibraciones, ocasionando una gran suma de ventajas y beneficios.

En primer lugar, dar especial importancia a la mejora de los riesgos laborales ya que se va aumentar rigurosamente la precisión y fiabilidad de los rodamientos mecánicos, siguiendo de manera fiel el comportamiento esperado y produciéndose menores desviaciones y con un error más reducido.

En segundo lugar, poder integrar en una única herramienta gran parte de las aplicaciones necesarias para poder llevar a cabo un estudio pormenorizado aplicado al mantenimiento predictivo de las vibraciones mecánicas que se generan en los rodamientos y así facilitar el trabajo del estudio de señales a los usuarios.

Por último, los niveles de rendimiento y productividad van a incrementarse significativamente ya que al poder trabajar en tiempo real se puede comprobar fácilmente el tiempo que tardan en producirse la estabilización de los rodamientos y así poder trabajar con mayor seguridad, durabilidad y confort en los trabajos.

Todos estás mejoras, tanto de las aplicaciones software e interfaz gráfica externa como todas las ventajas y beneficios que conllevan, hacen de esta herramienta una gran apuesta de futuro y desarrollo y son las principales y más importantes motivaciones que llevan a trabajar con ilusión y esmero en este Proyecto de Fin de Carrera.

1.2 Objetivos

El objetivo del presente Proyecto de Fin de Carrera es el desarrollo en MATLAB de una interfaz gráfica de guiado que permita las fases de captura, procesamiento y monitorización de señales destinadas a la detección de fallos en rodamientos y ejes.

De este modo, permite el estudio de señales tomadas desde una MFS Lite para medir y convertir las vibraciones mecánicas en señales eléctricas para su análisis.

A continuación, se van a describir los objetivos específicos:

- Creación de una interfaz gráfica externa mediante la utilización del programa MATLAB. Ésta posee la herramienta GUIDE (Graphical Use Interface Development Environment) que permite crear interfaces gráficas y dispone de las características básicas de todos los programas visuales y de interacción hombre-máquina. Estas son las llamadas GUI (Guide User Interface).

- Diseño de un panel de control para la interfaz gráfica externa lo más explícito e intuitivo posible para que el usuario que use la herramienta se adapte con facilidad al control y utilización de la misma.

- Creación de varios menús implícitos en el panel de control principal de la interfaz "Btool", que permitan elegir al usuario entre los 3 modos de funcionamiento previamente, tarjeta KEITHLEY, micrófono o simulación, antes de saltar a la interfaz de tratamiento de señal "Allchannels".

- Con la única ayuda de un PC y una tarjeta de adquisición de datos, realizar mediciones de cualquier magnitud, previa conversión a estímulos electrónicos, representándolas en función del tiempo para posteriormente poder tratarlas y llevar a cabo diversas transformaciones de las mismas con el objeto de realizar un estudio completo de las evoluciones y características particulares de éstas.

- La interfaz en modo simulación debe servir como banco de pruebas y aprendizaje para formar previamente a los usuarios antes de la captación real de señales. Este sistema debe permitir el tratamiento de las señales simuladas del mismo modo que las señales reales.

- Registro tanto de los datos de las señales reales adquiridas como de las simuladas para su posterior análisis mediante la creación de ficheros de almacenamiento específicos.

- Permitir realizar la carga de los datos en la interfaz principal “Btool” que se hayan registrado previamente en cualquiera de los 3 modos de funcionamiento.

- Medición de varias señales simultáneamente y en tiempo real en un rango comprendido entre uno y ocho canales a elegir por el usuario para mejorar las posibilidades de estudio de la señal e incrementar las prestaciones del software desarrollado.

1.3 Partes del documento

En el capítulo introductorio se explica al lector todas las motivaciones y necesidades que han existido para finalmente dar origen al presente Proyecto Fin de Carrera. También se hace referencia a todos los objetivos que debe cumplir el proyecto para lo que ha sido diseñado y finalmente se adjunta un apartado explicativo relativo a la estructura del proyecto donde se cita un resumen acerca de las diferentes partes del documento.

A continuación, se efectúa la primera toma de contacto del usuario con el trabajo realizado. Se realiza una explicación teórica sobre los temas más relevantes y concernientes para la correcta asimilación y deducción del documento, teniendo especial consideración por todo aquel usuario que no disponga de los conocimientos básicos necesarios para la compresión del mismo. Se realiza un estudio sobre la importancia del mantenimiento predictivo en los rodamientos mecánicos buscando mejorar el diseño y prestaciones de la aplicación futura a desarrollar para satisfacer de manera más óptima y eficiente su objetivo final.

Acto seguido, se explican y desarrollan los métodos utilizados para la programación y la adquisición de datos de la herramienta a desarrollar. Se expone el entorno de programación elegido para la programación del software, MATLAB, y se detallan las características y el funcionamiento de la aplicación a utilizar para la creación de interfaces gráficas, GUIDE. Además se analizan las diversas formas y dispositivos para la adquisición de señales y su respectiva comunicación y transmisión de los datos capturados al ordenador.

Posteriormente, se realiza una explicación pormenorizada de la interfaz gráfica a desarrollar, es decir, se describe la herramienta desde lo general a lo particular. Por tanto, se comienza explicando cual son las especificaciones de diseño de la herramienta detallando tanto sus requisitos de sistema como sus requisitos funcionales. Acto seguido, se analiza la arquitectura que el sistema va a desarrollar entre los diferentes dispositivos. Se estudian las formas de comunicación y transmisión de datos entre los diferentes módulos y modos de trabajo. Por último, ser realiza una descripción del sistema y se analizan las partes y funciones más significativas de la programación del software.

Antes de finalizar, se describen los resultados experimentales en los que se muestra el funcionamiento del sistema bajo circunstancias de trabajo de campo, demostrando el correcto funcionamiento de la aplicación y los resultados obtenidos como fruto de la prueba del mismo.

Por último, se ofrecen unas directrices a modo de sugerencia acerca de los futuros desarrollos que podría tener la herramienta en particular y el campo del mantenimiento predictivo en general, a la vista de los resultados obtenidos.

2. Estado del Arte

A continuación se introducirá al lector en las diferentes disciplinas que abarca este proyecto, presentándole para cada una de ellas los conceptos básicos, generalidades, clasificaciones, diseños de interés y explicación detallada para cada apartado expuesto, con el fin de que la lectura del Proyecto de Fin de Carrera pueda realizarse sin dificultad, incluso para aquellos que nunca hayan tratado con los temas aquí abordados. Las disciplinas que se revisarán son las siguientes: mantenimiento predictivo, procesamiento de señales vibratorias, técnicas de análisis espectral de señales, interfaces gráficas y entornos de programación gráfica.

2.1 Mantenimiento predictivo

2.1.1 Introducción

El mantenimiento de los componentes de las máquinas ha sido uno de los campos más importantes de la ingeniería mecánica a lo largo de su historia. La correcta evaluación del estado dinámico de dichos componentes es el elemento clave para un correcto plan de mantenimiento y en este campo es en el que se va a ver ubicado el presente proyecto.

Por norma general se clasifican los tipos de mantenimiento bajo los siguientes tres criterios [13]:

- **Mantenimiento correctivo:** Es aquel en el que se sustituyen las piezas dañadas en el mismo momento en que éstas fallan.
- **Mantenimiento preventivo:** Es aquel en el que se sustituyen las piezas cada cierto tiempo, independientemente de que éstas estén o no dañadas. La principal ventaja de este mantenimiento frente al correctivo es que se evitan las paradas imprevistas y con ello pérdidas por paradas de producción, mientras que la principal desventaja es que muchas veces se desperdician piezas que podrían haber seguido funcionando durante más tiempo del que lo han hecho.
- **Mantenimiento predictivo:** Es aquel en el que se llevan a cabo ciertas medidas para tratar de adivinar cuándo un elemento va a fallar y así, programar una parada para su sustitución cuando se considere oportuno.

Obviamente, estas tres medidas no aparecieron al mismo tiempo, sino que son el resultado de una evolución temporal que buscaba la mayor eficiencia posible en la producción [6].

En las últimas décadas, las estrictas normas de calidad y la presión competitiva han obligado a las empresas a transformar sus departamentos de mantenimiento, cobrando con ello cada vez una relevancia mayor a medida que se avanzaba en el campo.

Por eso en la situación actual es imprescindible, tanto en las grandes como en las medianas empresas, la implantación de una estrategia de mantenimiento adecuada que consiga aumentar la vida de sus componentes, mejorando así la disponibilidad de sus equipos y su confiabilidad, lo que repercute en una mayor productividad de la planta.

La gestión del mantenimiento ha evolucionado mucho a lo largo de un espacio de tiempo relativamente corto. Es por ello por lo que el mantenimiento industrial, día a día, está rompiendo con las barreras del pasado a pasos agigantados.

Actualmente, muchas empresas aplican la frase "el mantenimiento es inversión, no gasto" como una de sus más importantes consignas a la hora de planificar el funcionamiento de las mismas.

El primer tipo de mantenimiento llevado a cabo por las empresas fue el llamado mantenimiento correctivo, también llamado mantenimiento de emergencia. Esto era debido a que esta clase de mantenimiento se basa únicamente en solucionar los problemas de los equipos cuando fallan, reparando o sustituyendo las piezas o equipos estropeados.

Estas técnicas quedaron rápidamente obsoletas, ya que, si bien el programa de mantenimiento está centrado en solucionar el fallo cuando se produce, va a implicar por otro lado altos costes debido al descenso de la productividad y a las mermas en la calidad que originará.

De esta situación surge el mantenimiento preventivo, cuyo objetivo es evaluar la condición de salud de la máquina y su evolución en el tiempo mientras está funcionando, es decir, se trata de intentar anticiparse a la aparición de fallos en el tiempo. Esto evita detener la máquina debido a una avería y su consiguiente reparación lo que sin duda supone un importante ahorro del tiempo no productivo.

Para determinar la condición de la máquina se evalúa una serie de síntomas que la misma emite al exterior. Por esta razón, a esta estrategia de mantenimiento también se le conoce como mantenimiento sintomático.

Dentro de los síntomas que se analizan hay una gran variedad de ellos. Sin embargo, en la mayoría de las empresas el mantenimiento predictivo está centrado en el análisis de vibraciones junto con otras técnicas complementarias como el análisis de aceites, termografía, ultrasonido, etc.

2.1.2 Situación actual

El concepto de mantenimiento predictivo o mantenimiento basado en condiciones comenzó a ser aplicado por la industria a principios de la década de los ochenta. Aparecen por aquella época los primeros equipos portátiles tipo "colector de datos" con memoria interna para la medición periódica en planta de las condiciones en las que se encontraban los equipos que componían la planta de producción, con interfaces de conexión a los primeros PCs disponibles en el mercado[6].

Obviamente, se trataba de un sistema muy primitivo aún que estaba en sus primeros pasos camino al desarrollo que permitiera que la estrategia de mantenimiento preventivo fuera llamativa para el público masificado. Por este motivo, solamente hasta bien entrados los años noventa, se observa que el mantenimiento predictivo comienza a jugar un papel importante dentro de la industria.

Las predicciones comienzan a ser cada vez más precisas, suponiendo con ello un ahorro creciente a lo largo del tiempo, pues las condiciones de predicción no paraban de mejorar y mantenían los visos de continuar haciéndolo en un futuro cercano. Las direcciones comienzan a identificar la ventaja y necesidad de invertir en el desarrollo de un programa de mantenimiento predictivo de forma cada vez más generalizada. Se pasó de ser un tipo de mantenimiento empleado por unos cuantos pioneros a ser la estrategia adoptada por cada vez más empresas, interesadas en reducir los costes de producción mediante este sistema.

La situación actual nos habla de una aplicación masificada del mantenimiento predictivo, ya que hace tiempo se asumió que era la mejor técnica para la detección de fallos con la maquinaria aún en funcionamiento.

No obstante, no es un campo que esté cerrado en este sentido, ya que se sigue investigando para conseguir que la tasa de acierto a la hora de detectar los posibles fallos de los equipos sea lo más elevada posible. Los campos por excelencia en los cuales se aplica esta estrategia son aquellos procesos de producción que se caracterizan por una gran repetición de procesos, estando las operaciones que se realizan en la planta, muy encorsetadas, y en las cuales la maquinaria trabaja durante un gran número de horas ininterrumpidamente.

El ejemplo más claro de este tipo de procesos sin duda es el de la industria del automóvil y es debido a que esta industria fue una de las pioneras en el mantenimiento predictivo y una de las que más invierte en la mejora del mismo, ya que una eventual parada en el sistema de producción acarrea pérdidas muy elevadas que no se pueden tolerar con frecuencia si se quiere que el sistema sea rentable.

2.1.3 Aplicación y metodología

El uso del mantenimiento predictivo consiste en establecer, en primer lugar, una perspectiva histórica de la relación entre la variable seleccionada y la vida del componente. Esto se logra mediante la toma de lecturas de datos procedentes de una fuente de información tal como, por ejemplo, la vibración de un cojinete, en intervalos periódicos hasta que el componente se rompa o se averíe, recogiendo y estudiando posteriormente la previa lectura de los datos obtenidos. En función del estudio de dichos datos se deberá determinar si compensa o no aplicar la estrategia de mantenimiento predictivo, atendiendo a distintas variables, como por ejemplo el coste del elemento a reemplazar, el tiempo de reparación durante el cual la maquinaria ha de estar detenida y con ella la producción, el coste que va a suponer la recogida de información y su tratamiento, etc [13].

Una vez determinada la factibilidad y conveniencia de realizar o no un mantenimiento predictivo a una máquina o línea, el siguiente paso es determinar las variables físicas a controlar que sean indicativas de la condición de la máquina. Como se puede deducir, esta decisión es primordial y determinante a la hora de obtener un resultado satisfactorio en el mantenimiento.

El objetivo es revisar en forma detallada las técnicas comúnmente usadas en la monitorización según condición de la planta de producción, de manera que sirvan de guía para la selección, puesto que basándose en la experiencia que nos ofrecen casos previos similares al propio, se estará probando una tecnología testada, lo que eleva las posibilidades de éxito considerablemente.

La finalidad de la monitorización de los datos es obtener una indicación de la condición mecánica o estado de salud de la máquina, de manera que pueda ser operada y mantenida con seguridad y eficacia. De acuerdo a los objetivos que se pretenden alcanzar con la monitorización de la condición de una máquina debe distinguirse entre vigilancia, protección y diagnóstico [6].

- **Vigilancia de máquinas:** Su objetivo es indicar cuándo existe un problema. Debe distinguir entre condición buena y mala, y si es mala indicar su grado de severidad.
- **Protección de máquinas:** Su objetivo es evitar averías catastróficas. Una máquina está protegida, si cuando los valores que indican su condición llegan a valores considerados peligrosos, la máquina se detiene automáticamente.
- **Diagnóstico de averías:** Su objetivo es definir cuál es el problema específico. Su objetivo es estimar cuánto tiempo más podrá funcionar la máquina sin riesgo de sufrir una avería.

2.1.4 Técnicas aplicadas al mantenimiento predictivo

Existen varias técnicas aplicadas para el mantenimiento preventivo entre las cuales destacan las siguientes [6]:

- Análisis de vibraciones.
- Ondas de lubricante.
- Análisis de ondas de alta frecuencia.
- Análisis por termografía.
- Análisis por corriente eléctrica.

Análisis de vibraciones

El interés de las vibraciones mecánicas llega al mantenimiento industrial de la mano del mantenimiento preventivo y predictivo, debido al interés de alerta que implica un elemento vibrante en una máquina, debido a su peligrosidad y al deterioro que origina en la misma y la necesaria prevención de las averías que provocan las vibraciones a medio plazo.

Es por este motivo por el que será el tipo de análisis en el que se centra el proyecto, debido a su gran importancia y aplicación en el panorama industrial.

El hecho de que la distribución de cargas varíe con el tiempo causa que los elementos mecánicos se comporten como generadores de vibraciones, incluso aunque estos no tengan ningún defecto.

Sin embargo, la presencia de defectos hace que ciertas frecuencias se amplifiquen o que aparezcan nuevas. Véase *Figura* 1.

Los defectos originados por vibraciones pueden clasificarse como localizados y distribuidos [13]:

- **Defectos localizados:** Los más comunes son aquellos que son causados por la propagación de fisuras hacia la superficie debido a la fatiga. El fallo por fatiga se ve favorecido cuando el elemento está sobrecargado o soporta cargas de impacto durante su funcionamiento o instalación.
- **Defectos distribuidos:** Tres de los ejemplos más extendidos son la rugosidad superficial, las pistas desalineadas y los elementos rodantes desiguales (para el caso de los rodamientos). Tienen su origen por norma general en errores de fabricación o una inadecuada instalación.

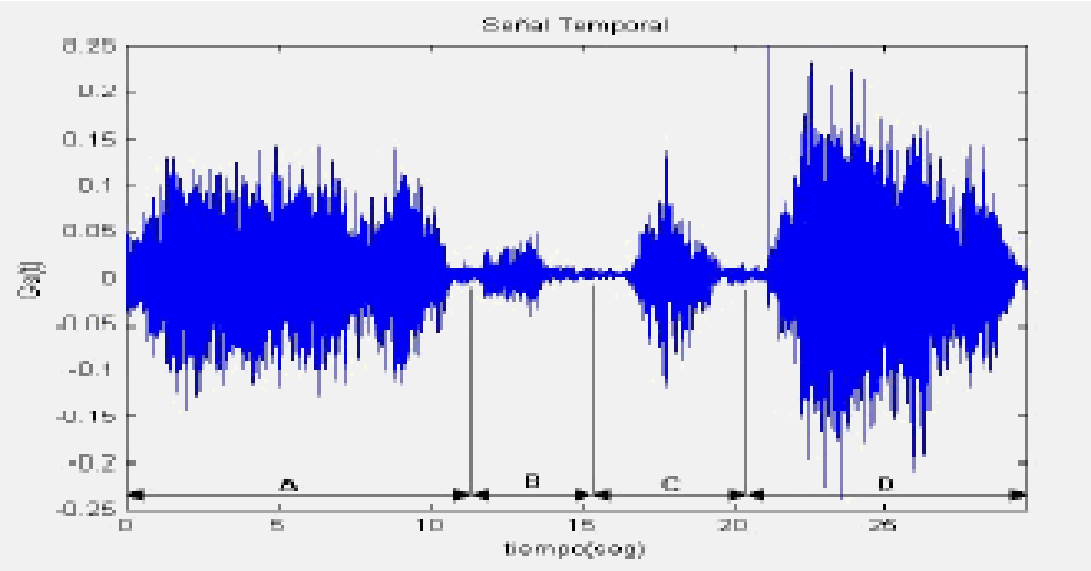


Figura 1. Registro de las vibraciones en un ciclo de trabajo en función del tiempo

El interés principal para el mantenimiento deberá ser la identificación de las amplitudes predominantes de las vibraciones detectadas en el elemento o máquina, la determinación de las causas de la vibración y la corrección del problema que ellas representan.

Las consecuencias de las vibraciones mecánicas son el aumento de los esfuerzos y las tensiones, pérdidas de energía, desgaste de materiales y las más temidas: daños por fatiga de los materiales, además de ruidos molestos en el ambiente laboral.

Entre las razones más habituales por las que una máquina o elemento de la misma puede llegar a vibrar cabe destacar [6]:

- **Desequilibrio:** Todos los elementos rotativos son siempre fuentes potenciales de vibraciones mecánicas. El desequilibrio en la distribución de la masa es una de las causas más comunes ya que, únicamente cuando el eje de giro coincida con el de gravedad, las fuerzas de inercia no producirán ninguna acción centrífuga perturbadora en los rodamientos. En el mundo real esto es imposible de conseguir por muy estrictas que sean las tolerancias de fabricación; es por ello por lo que siempre se tendrá un cierto grado de desequilibrio en las máquinas rotativas.

- **Desalineamiento:** El desalineamiento produce una pequeña sobrecarga que puede ocasionar la ruptura de la película de aceite con el consiguiente riesgo de falta de lubricación en la zona de carga. En general, cualquier sobrecarga por desalineamiento reduce la vida útil de un rodamiento.

- **Corrosión:** Es un proceso químico que experimentan la mayoría de los metales y que contribuye a su deterioro. Por este motivo es una variable más importante que la misma corrosión, la velocidad con la que se da este proceso.

- **Excentricidad:** La excentricidad es otra de las causas comunes de vibración en la maquinaria rotativa. Significa que la línea central del eje no es la misma que la línea central del rotor, con los consiguientes problemas que acarreará.

- **Defectos en rodamientos, cojinetes, engranajes o correas:** Suelen venir determinados por errores de fabricación o de instalación en el sistema. Acortan la vida de los elementos y provocan vibraciones.

- **Holguras:** Se trata de un espacio donde el cual el engranaje se mueve de manera incontrolada, provocando choques indeseados que acarrean vibraciones entre los elementos mecánicos. Una vez que aparecen, suelen ir acentuándose hasta desembocar en una rotura del elemento que la sufre.

- **Falta de lubricación:** Cerca del 36% de los fallos prematuros de los rodamientos son causados por especificaciones y aplicaciones incorrectas de los lubricantes. Inevitablemente, cualquier rodamiento sin una correcta lubricación fallará antes de llegar a su vida nominal de servicio. Los rodamientos son a menudo uno de los componentes de más difícil acceso en la maquinaria, por ello, una lubricación inadecuada supone habitualmente problemas complejos.

Una vez expuesto el marco en el que se va a centrar el presente proyecto, se pasa a describir los otros análisis que se pueden llevar a cabo para el estudio del mantenimiento predictivo.

Análisis por lubricante

Estos se ejecutan dependiendo de la necesidad, según tres criterios que a continuación se pasa a detallar:

- **Análisis Iniciales:** Se realizan a productos de aquellos equipos que presenten dudas provenientes de los resultados del estudio de lubricación y permiten correcciones en la selección del producto, motivadas por cambios en condiciones de operación.

- **Análisis Rutinarios:** Se aplican para equipos considerados como críticos o de gran capacidad, en los cuales se define una frecuencia de muestreo, siendo el objetivo principal de los análisis la determinación del estado del aceite, nivel de desgaste y contaminación entre otros.

- **Análisis de emergencia:** Se efectúan para detectar cualquier anomalía en el equipo y/o lubricante, como por ejemplo la contaminación del mismo con agua, la presencia de partículas sólidas, provenientes de filtros y sellos defectuosos o el uso de un producto lubricante inadecuado atendiendo a las especificaciones de la máquina. Véase *Figura 2*.

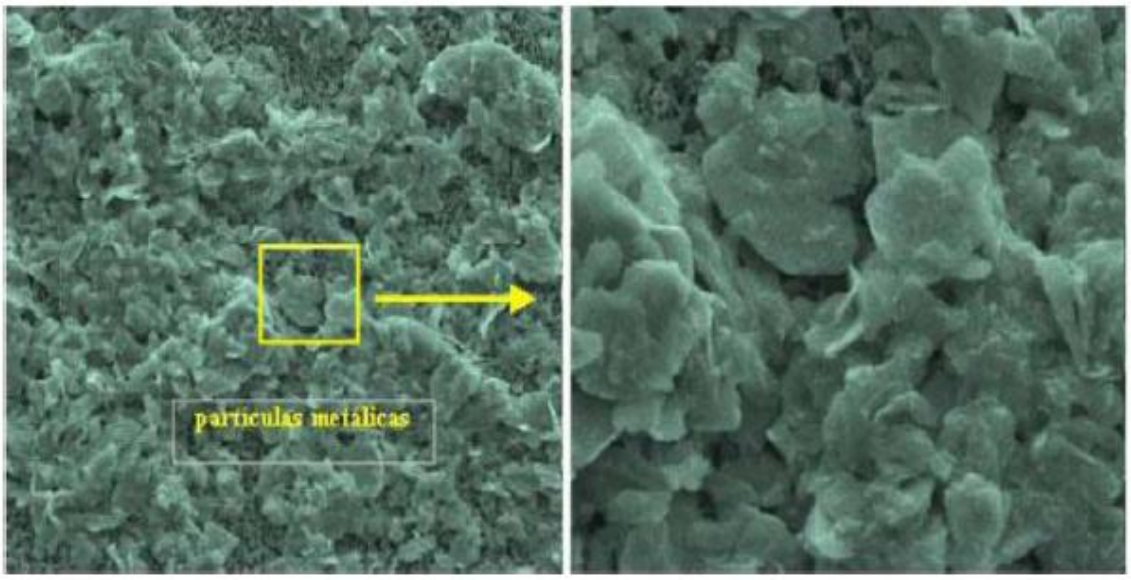


Figura 2. Presencia de partículas sólidas en el análisis de aceites

Análisis por ondas de alta frecuencia

Este método estudia las ondas de sonido de alta frecuencia producidas por los equipos que no son perceptibles por el oído humano (ultrasonidos). Véase *Figura 3.*

Figura 3. Análisis por ultrasonidos de un sistema

Casi todas las fricciones mecánicas, arcos eléctricos y fugas de presión o vacío producen ultrasonido en un rango aproximado a los 40 KHz. Estas son frecuencias con características muy aprovechables en el mantenimiento predictivo, puesto que las ondas sonoras son de corta longitud atenuándose rápidamente sin producir rebotes. La alta direccionalidad del ultrasonido en 40 KHz permite localizar con rapidez y precisión la ubicación del defecto.

Además de ésta aplicación, el análisis por ultrasonidos tiene estos otros campos de aplicación, ya que los ultrasonidos permiten detectar [7]:

- Detección de fricción en máquinas rotativas.
- Detección de fallas y/o fugas en válvulas.
- Detección de fugas de fluidos.
- Pérdidas de vacío.
- Detección de "arco eléctrico".
- Verificación de la integridad de juntas de recintos estancos.

Análisis por termografía

La termografía infrarroja es una técnica que permite, a distancia y sin ningún contacto, medir y visualizar temperaturas de superficie con precisión. Los ojos humanos no son sensibles a la radiación infrarroja emitida por un objeto, pero las cámaras termográficas son capaces de medir la energía con sensores infrarrojos, capacitados para "ver" en estas longitudes de onda. Esto permite medir la energía radiante emitida por los objetos y, por consiguiente, determinar la temperatura de la superficie a distancia, en tiempo real y sin contacto.

La gran mayoría de los problemas y averías en el entorno industrial, ya sea de tipo mecánico, eléctrico y de fabricación, están precedidos por cambios de temperatura que pueden ser detectados mediante la monitorización de temperatura con sistema de termografía por infrarrojos.

Con la implementación de programas de inspecciones termográficas en instalaciones, maquinaria o cuadros eléctricos, entre otros, es posible minimizar el riesgo de una avería de equipos y sus consecuencias, a la vez que también ofrece una herramienta para el control de calidad de las reparaciones efectuadas.

El análisis mediante termografía infrarroja debe complementarse con otras técnicas y sistemas de ensayo conocidos como pueden ser el análisis de lubricantes, el análisis de vibraciones, los ultrasonidos pasivos y el análisis predictivo en motores eléctricos. Véase figura *Figura 4*.

Se puede observar una termografía de un cuadro de mando de una instalación eléctrica. Los colores más claros, tales como el rojo, naranja o amarillo indican las temperaturas más elevadas dentro de dicho análisis mientras que los más oscuros, como el morado o el negro incluso señalan las zonas con menores valores de temperatura.

La información facilitada por el estudio en este caso indica los conectores que están sufriendo un mayor desgaste por exposición a temperaturas elevadas, pudiendo deberse el mismo a una mayor utilización respecto al resto de los mismos o a una mala conexión o estado de la instalación que se conecta a dichos terminales.

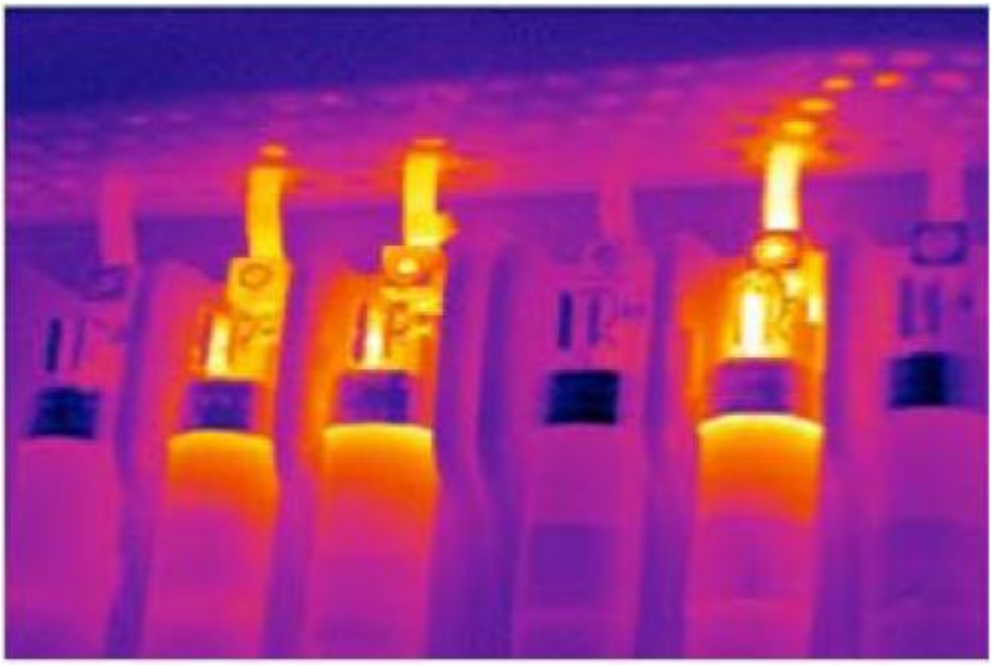


Figura 4. Imagen obtenida por termografía de un cuadro de mando

Análisis por corriente eléctrica

El objetivo del análisis eléctrico como técnica de mantenimiento predictivo es el de realizar estudios eléctricos sobre aquellos equipos que pueden presentar averías de origen electro-mecánico y obtener conclusiones en cuanto a la peligrosidad de la exposición a las mismas para el equipo que se estudie [6]. Véase *Figura 5*.

En función de la corriente de alimentación, trifásica o continua, del equipo que se desea analizar, se pueden verificar las siguientes condiciones:

- Estado del circuito.
- Estado del aislamiento.
- Estado del estator.
- Estado del rotor.
- Excentricidades en el entrehierro.

Estas cinco condiciones son las más extendidas, pero existen muchas otras en función de las especificaciones del sistema y del entorno en el que se va a ubicar.

Es tarea del responsable del análisis por corriente eléctrica determinar los factores más relevantes que se han de someter a estudio para obtener de esta forma la información más útil para los objetivos que se estén persiguiendo para cada situación.

Además de lo expuesto, el análisis de corriente de un motor eléctrico puede desempeñarse a modo de control de calidad, como herramienta de tendencia o como emisor de un diagnóstico inmediato del estado del mismo, por lo que se ha de conocer adecuadamente el abanico de posibilidades y el equipo en cuestión que se vaya a analizar para poder así obtener un diagnóstico correcto.

Figura 5. Técnico llevando a cabo un análisis por corriente eléctrica

2.2 Procesamiento de señales vibratorias

2.2.1 Introducción

El análisis de vibraciones es una de las prácticas más usadas dentro del conjunto del mantenimiento predictivo. Requiere un conocimiento de las señales y su análisis es uno de los pilares fundamentales de la ingeniería. La información que contienen las señales debe transformarse dependiendo de los propósitos de estudio para poder así obtener la información más adecuada de acuerdo con las premisas marcadas [8].

En esta última década, se ha realizado un esfuerzo investigador notable para desarrollar técnicas de detección y diagnosis basadas en medidas vibratorias. Estas técnicas se pueden aplicar en el dominio temporal, en el dominio de la frecuencia o en el dominio tiempo-frecuencia.

Los análisis más sencillos son aquellos basados en medidas temporales. Estos sistemas emplean habitualmente medidas estadísticas efectuadas sobre las historias temporales, con el fin de establecer parámetros de tendencia que permitan de detectar la presencia de un modo de fallo.

El uso del dominio de la frecuencia se impone frente al dominio del tiempo, a pesar de su aparente mayor complejidad debido a un conjunto de razones entre las que destacan las siguientes:

- El significado físico es a menudo más fácil de obtener en el dominio de la frecuencia que en el del tiempo en la descripción de señales y sistemas.
- Los patrones significativos de la señal son, a menudo, más fáciles de reconocer. Pequeños cambios en la señal son reconocidos inmediatamente en la representación del dominio de la frecuencia.
- Los sistemas mecánicos se modelan frecuentemente mediante un sistema lineal, descrito por ecuaciones diferenciales lineales. Mediante el uso de la transformada de Fourier se pueden convertir dichas ecuaciones diferenciales en algebraicas, con la consiguiente mejora a la hora de trabajar con ellas.

Las técnicas basadas en análisis realizados en frecuencia utilizan como rasgos fundamentales para fijar medidas de tendencia las amplitudes de los armónicos dominantes en la respuesta, así como los anchos de banda asociados.

Sin embargo, este tipo de análisis no es capaz de detectar fallos locales ya que la transitoriedad de estos eventos en el dominio temporal queda enmascarada en el espectro obtenido al realizar la transformación en frecuencia.

Por consiguiente, el seguimiento tanto de los anchos de banda como de los armónicos afectados se ve, en caso de fallos locales, seriamente dificultado.

Una alternativa para resolver este problema se encuentra en el empleo de análisis tiempo-frecuencia, los cuales ofrecen una medida de la distribución de energía de la señal en ambos dominios simultáneamente, pero con distinta resolución de acuerdo con el principio de incertidumbre de Heisenberg.

Dentro de estas técnicas se encuentran la transformada corta de Fourier (STFT, del Inglés Short Time Fourier Transform), las distribuciones Wigner-Ville y Choi-Williams y, por último, la transformada Wavelet. Esta última ha alcanzado bastante auge en la última década, por su capacidad para trabajar con transitorios y periodicidades.

2.2.2 Clasificación de las señales vibratorias

Desde el punto de vista de la cinemática, ver *Figura 6*, las señales en general se dividen en dos grandes grupos:

- **Determinísticas:** Son aquellas señales que representan fenómenos que pueden ser descritos analíticamente de manera exacta mediante una expresión matemática relativamente sencilla. Se dividen en otros dos grupos.

  - Periódicas: Son aquellas señales que repiten su comportamiento cada cierto intervalo de tiempo fijo.

  - No periódicas: Se trata de señales que no repiten su comportamiento.

- **Aleatorias o estocásticas:** Son señales que no es posible describir analíticamente con una expresión explícita simple como en el caso anterior.

Sin embargo, cuando una señal estocástica se observa durante un largo periodo de tiempo puede verse cierta regularidad y puede ser descrita en términos de probabilidades y promedios estadísticos. Se subdividen en dos grupos:

- **Estacionarias:** Las señales estacionarias son constantes en sus parámetros estadísticos a lo largo del tiempo. Si se observa una señal estacionaria en un momento dado y se repite la observación transcurrido un determinado periodo de tiempo aleatorio, esencialmente se observaría lo mismo, es decir, su nivel general, su distribución de amplitud y su desviación típica tomarían los mismos valores en ambos casos. La maquinaria rotativa generalmente produce señales de vibración estacionarias.

- **No estacionarias:** Son aquellas que no mantienen su valor a lo largo del tiempo, ya que van variando y por ello, cada vez que se las observe se percibirá un valor de la señal diferente.

Por lo que la clasificación de modo gráfico sería la siguiente [4]. Véase *Figura 6*.

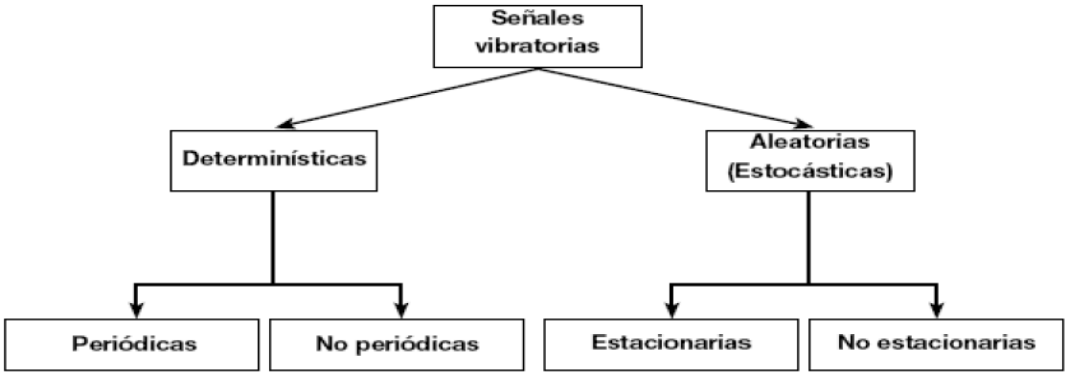


Figura 6. Clasificación de las señales vibratorias

2.2.3 Modelo de procesamiento de señales vibratorias

Para estudiar mejor las diferentes causas de las vibraciones en los sistemas, se ha de crear modelos dinámicos y matemáticos simples que los describan.

Sobre el comportamiento vibratorio de los sistemas influyen tanto sus características constructivas como sus características de trabajo. Las vibraciones en los sistemas serán reflejo de su comportamiento dinámico.

El modo de actuar ante el estudio y procesamiento de señales vibratorias es el presentado a continuación. Véase *Figura 7*.

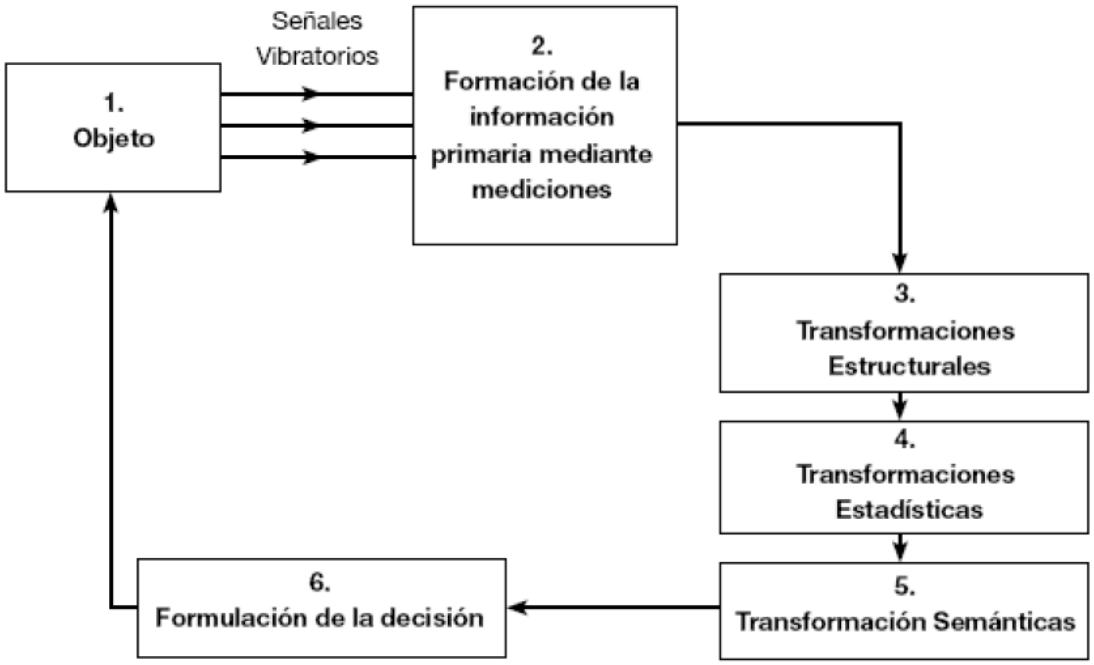


Figura 7. Metodología para el procesamiento de señales vibratorias

La información es una categoría abstracta en principio pero que en el caso a tratar se presenta en forma de señales.

La señal es la variación del parámetro físico que vaya a ser objeto de estudio por parte del técnico, en función del tiempo y su función es la de transportar cierta cantidad de información desde un emisor, que es la máquina objeto de estudio (1) hasta un receptor, que en este caso se trata de aparatos de medición a través de los cuales se ejerce la primera transformación de las señales obtenidas (2).

El ruido en las señales se separa en las etapas 3, 4 y 5 mediante tres tipos de transformaciones, que son las transformaciones estructurales, estadísticas y semánticas, a través de métodos técnicos que finalmente desembocan en la formulación de la decisión, que es la información desde la cual obtener las conclusiones del estudio.

2.2.4 Tipos de análisis de señales vibratorias

2.2.4.1 Nivel básico

Análisis en frecuencia

El análisis espectral es una herramienta clásica empleada en el análisis de fallos. Los primeros analizadores de espectros fueron analógicos y aparecieron en los años 60, aunque todavía hay empresas que los siguen utilizando. Véase *Figura 8*.

Sin embargo, esto se masificó en las empresas con la aparición de los recolectores analizadores digitales a inicios de los años 80 [14].

Figura 8. Analizador de espectros analógico

La esencia del análisis espectral es descomponer la señal vibratoria medida con un sensor de vibraciones en sus componentes espectrales en frecuencia. Esto permite en el caso de las máquinas, correlacionar las vibraciones medidas generalmente en sus descansos, con las fuerzas dinámicas que actúan dentro de ella.

El análisis espectral consiste fundamentalmente en la transformada inversa de Fourier del logaritmo del espectro de potencia a fin de destacar las periodicidades que se hayan registrado en la medida vibratoria.

Este tipo de análisis puede ser útil cuando la máquina rotativa a analizar no contenga demasiados pasos de reducción, sin embargo su aplicación a máquinas más complejas deja de ser eficiente por el número de componentes a analizar; además, tiene el inconveniente de no indicar la localización del defecto.

Análisis de la forma de onda o de la vibración

El análisis de la forma de la vibración u onda en el tiempo es un análisis complementario al análisis de espectros y para detectar algunos problemas específicos como impactos y transitorios es más efectivo que el anterior. Véase *Figura 9.*

Sin embargo, en una gran parte de problemas deberían ser usados integradamente.

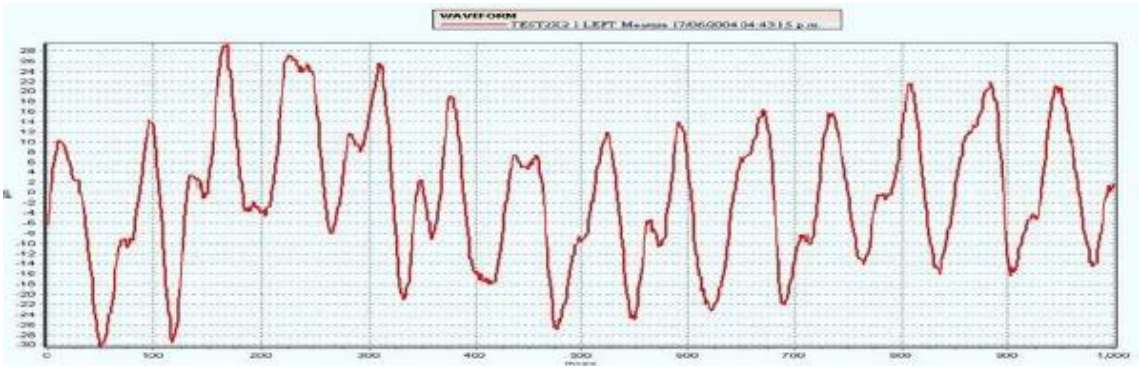


Figura 9. Análisis de la forma de onda de una vibración

Análisis de la diferencia de fase de vibraciones

La diferencia de fase entre dos vibraciones de igual frecuencia se puede definir como la diferencia en tiempo o en grados con que ellas llegan a sus valores máximos, mínimos o cero.

La fase de las vibraciones se mide normalmente respecto a un pulso de referencia obtenido de un fototacómetro. Véase *Figura 10*.

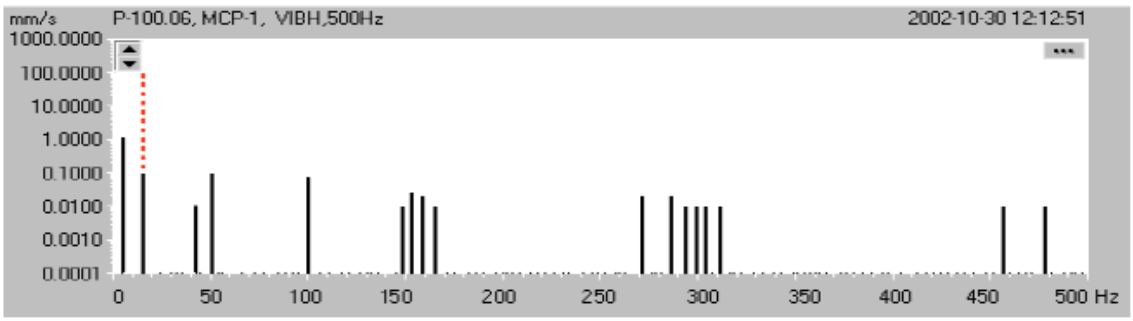


Figura 10. Vibraciones a distintas frecuencias

2.2.4.2 *Nivel medio. Análisis temporales y estadísticos*

Se basan en la comparación de promedios síncronos (en determinado régimen de vueltas de la maquinaria) de una señal de referencia obtenida a priori cuando el par cinemático de engrane esta "sano" (se tiene la certeza de que está sin ningún tipo de daño) frente a la señal obtenida en condiciones de funcionamiento de la máquina (funcionando en el mismo punto de operación) [14]. Véase *Figura 11*.

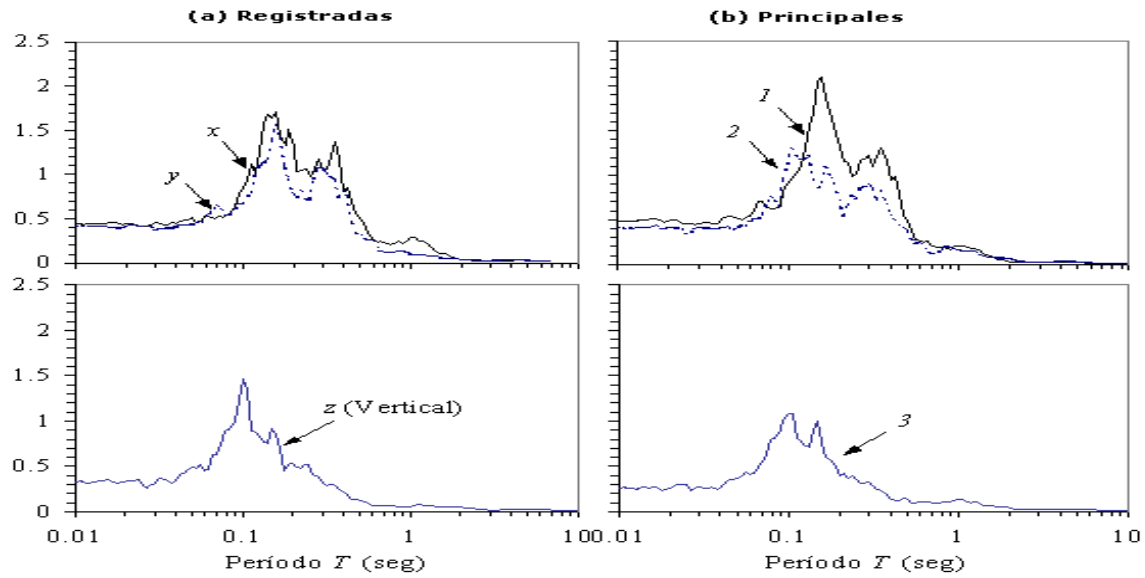


Figura 11. Variación temporal de las vibraciones

Existen a su vez dentro de este tipo de análisis las siguientes subcategorías:

Técnicas de demodulación de amplitud y fase del residuo

Esta técnica se basa en un filtrado de la señal "pura" (sin tratar) alrededor de la frecuencia de engrane y de sus armónicos (hay que seleccionar un ancho de banda).

Posteriormente se pasa a sustraer dicho armónico fundamental y recuperar la envolvente de la señal residual (tanto en amplitud como en fase) en el dominio temporal, con el objetivo de captar el efecto fundamental anteriormente comentado de modulación de la información. Véase *Figura 12*.

Hay que destacar que esta técnica es muy sensible a la selección del ancho de banda (si no se elige correctamente se puede llegar a perder cierta información de interés).

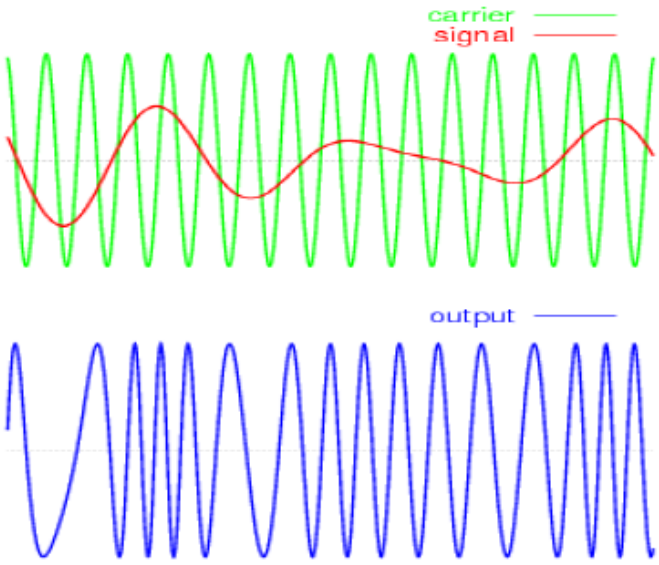


Figura 12. Modulación FM de señal mostrando las señales portadora y modulada

Análisis estadísticos

Se basan en realizar estadísticas sobre el residuo de la señal obtenida al sustraer la señal leída en condiciones operativas de la máquina respecto a la de referencia.

A grandes rasgos todas las técnicas arriba mencionadas tienen la ventaja de que al promediar se elimina parte del ruido "contaminante" de las señales; no obstante, hay que asegurar muy bien la sincronización puesto que de no ser así se podrían eliminar ciertos eventos periódicos que tuvieran relevancia en el dictamen de la diagnosis de la maquinaria rotativa.

Como ventaja cabe citar que los análisis temporales se han mostrado útiles tanto en la detección como en la localización del daño y además son sensibles a la evolución de la degradación.

Análisis de órbitas

La órbita es la forma como se mueve el centro del eje del rotor en un plano perpendicular a su eje. Véase *Figura 13*.

Se obtiene mediante la combinación de los desplazamientos vibratorios que tienen lugar, los cuales pasan a ser captados por dos sensores ubicados relativamente entre ellos a 90° (por ejemplo en las direcciones horizontal y vertical respectivamente, aunque puede haber otras ubicaciones).

Figura 13. Técnico llevando a cabo un análisis de órbitas

2.2.4.3 Nivel avanzado

Transformadas tiempo-frecuencia

El análisis espectral clásico es adecuado para analizar vibraciones compuestas de componentes estacionarias durante su periodo de análisis. Esto indica que si se producen efectos transitorios en la vibración ellos son promediados en el periodo de análisis, perdiéndose toda información sobre la naturaleza o forma de estas variaciones.

Existe entonces la necesidad de un análisis que describa mejor señales no estacionarias o transitorias. Esto se consigue con las distribuciones o transformadas tiempo-frecuencia.

Las transformadas tiempo-frecuencia son análisis tridimensionales amplitud-tiempo-frecuencia, es decir, se agrega una nueva dimensión, el tiempo, a la clásica FFT. Existen varios tipos de transformadas tiempo-frecuencia, las cuales se pueden clasificar en lineales y no lineales.

Dentro de las primeras las más conocidas son la transformada corta de Fourier (Short Time Fourier Transform) y la transformada Wavelet. Respecto a las no lineales están la pseudo-transformada de Wigner-Ville, la Choi-Williams. Estas transformadas se están implementando actualmente en algunos analizadores de vibraciones comerciales debido a que su uso no se aplica de forma directa como con la FFT (Fast Fourier Transform, que es la aplicación de la transformada de Fourier discreta optimizada para SW).

Se requiere para su uso gran conocimiento del usuario y dependiendo del problema a analizar es más útil usar una u otra transformada.

Análisis espectral

En el análisis espectral clásico cada componente espectral se asocia a una fuerza dinámica que la genera.

Este análisis es por lo tanto adecuado a vibraciones estacionarias. Cuando es aplicado a máquinas que trabajan a velocidad variable, las componentes espectrales se dispersan en el espectro haciendo imposible su análisis.

Este proceso se puede realizar a través de dos técnicas: muestreo directo por hardware y re-muestreo por software (se adquiere la vibración y un pulso de referencia de un tacómetro a intervalos de tiempo constante y luego por software se realiza el re-muestreo a intervalos de ángulo constante). Una vez realizado este proceso se aplica la tradicional FFT.

2.3 Técnicas de análisis espectral de señales

2.3.1 La transformada corta de Fourier y el espectograma

La transformada de Fourier (FT) es una herramienta matemática que sirve para describir el comportamiento de señales no periódicas. Su objetivo es la descomposición de la función en una suma de funciones armónicas [14].

La definición de dicha transformada viene dada por las siguientes ecuaciones:

$$x(t) = \int_{-\infty}^{+\infty} X(f) exp(j2\pi kf) df$$

$$x(t) = \int_{-\infty}^{+\infty} X(f) exp(-j2\pi kf) df$$

De manera simbólica se puede decir que:

X(f) es la transformada de Fourier de x(t), siendo X(t) la representación de la señal en el dominio del tiempo y X(f) la representación en el dominio de la frecuencia.

Cuando se trata con señales complejas compuestas por gran cantidad de armónicos, el análisis en el dominio de la frecuencia permite distinguir las frecuencias de los armónicos principales, labor que sería mucho más compleja si sólo se recurriera al análisis temporal.

2.3.2 La transformada corta de Fourier (STFT)

El uso de la transformada de Fourier a la hora de analizar señales no estacionarias como las que plantea el presente proyecto plantea un problema, ya que no es un caso aislado o específico, es más, casi la totalidad de las señales que se generan en el entorno industrial presentan un carácter no estacionario [14]. Es por ello por lo que dicho inconveniente toma una especial relevancia.

El problema radica en que el espectro de frecuencias puede no ser el mismo para distintos instantes de tiempo, consecuencia directa de la no estacionalidad de la señal con la que se está trabajando. Se tiene pues una existencia de numerosos espectros de frecuencia distintos dentro de una misma señal en función del instante de tiempo que se tome para el estudio.

Es por este motivo por el que si se representa la señal en el dominio de la frecuencia no se estará obteniendo información fidedigna de la misma. Para solucionar este problema surge la transformada corta de Fourier (en Inglés conocida como Short Time Fourier Transform, o STFT).

Este tipo de transformada pretende tratar la señal no estacionaria como un conjunto de señales adyacentes, consideradas como cuasi estacionarias.

Se tomará un intervalo de tiempo, dividiendo la señal en una serie de señales de este periodo, como si se tratase de una ventana de tiempo que se desliza a lo largo de la señal original.

Posteriormente se aplicará la FT a cada uno de estos intervalos, tal y como se muestra a continuación:

$$S_x(t,f) = \int_{-\infty}^{+\infty} x(u)h(u-t)exp(-j2\pi ft)du$$

En dicha ecuación x(t) es la señal a analizar y h(u – t) la ventana de tiempo que se traslada a lo largo de la señal.

El inconveniente que presenta la STFT es que tiene una resolución prefijada por el tamaño del intervalo, o lo que es lo mismo, el tamaño de la ventana temporal deslizante toma un valor fijo.

Si éste tiene una longitud infinita se obtendrá una representación en frecuencia perfecta a costa de perder toda la información temporal. Conforme el intervalo se hace menor se obtiene información en el dominio temporal, a costa de perder información en frecuencia, llegando al punto en el que no se conocen frecuencias concretas presentes en la señal sino bandas de frecuencia.

Este hecho se debe al principio de incertidumbre de Heisenberg, que postula lo siguiente:

"No se puede determinar, simultáneamente y con precisión arbitraria, ciertos pares de variables físicas, como son, por ejemplo, la posición y el momento lineal (cantidad de movimiento) de un objeto. En otras palabras, cuanto mayor certeza se busca en determinar la posición de una partícula, menos se conoce su cantidad de movimiento lineal, y por tanto, su velocidad".

Este principio puede aplicarse a ciertas parejas de variables físicas, si nos referimos al caso estudiado por el presente proyecto, se llegará a la conclusión de que no puede determinarse con exactitud en un cierto punto la información en el dominio temporal y en el de la frecuencia.

2.3.3 Espectograma

El módulo de la transformada corta de Fourier (STFT) vista en el apartado anterior elevado al cuadrado se denomina espectrograma y viene dado por:

El espectrograma puede ser representado en 3D (representación en cascada) o en 2D. Dicha representación tiene las mismas limitaciones que la STFT en cuanto a lo que el principio de incertidumbre de Heisenberg se refiere. Véase *Figura 14* y *Figura 15*.

Figura 14. Espectrograma en dos dimensiones

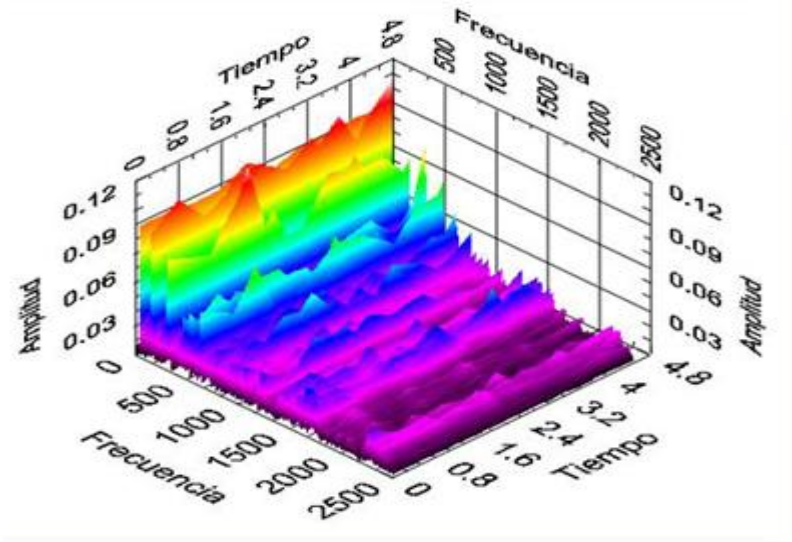


Figura 15. Espectrograma en tres dimensiones

En la *Figura 14* y *Figura 15*, se tienen sendas representaciones del espectrograma de una señal en 2D y en 3D respectivamente. En ellas se observa como a medida que el intervalo temporal se hace más pequeño, se obtiene información más precisa en el dominio temporal a costa de perder precisión en el dominio de la frecuencia y viceversa.

2.3.4 La transformada de Hilbert

Para superar las limitaciones impuestas por la transformada de Fourier y la transformada corta de Fourier surge la necesidad de encontrar un método para el análisis frecuencial de señales no lineales y no estacionarias.

En éste ámbito es donde entra en juego la transformada de Hilbert, que se muestra como una herramienta muy potente para estudiar este tipo de señales.

La expresión analítica de la transformada de Hilbert viene dada por la siguiente ecuación:

$$\eta\{s\}(t) = (h * s)(t) = \frac{1}{\pi}\int_{-\infty}^{+\infty} \frac{s(\tau)}{t-\tau} d\tau$$

En el marco del presente proyecto es importante haber definido ésta clase de transformada, ya que la aplicación "Btool" diseñada ofrece la posibilidad de calcular la transformada de Hilbert de las señales a tratar.

2.3.5 La transformada corta de Wavelet

Dado el principio de incertidumbre, lo mejor que se puede hacer es determinar las componentes espectrales existentes en un determinado intervalo temporal. La bondad de los resultados obtenidos depende de la resolución empleada en el análisis; es por ello por lo que se introduce el concepto de Wavelet, ya que proporcionará una resolución variable en el tiempo [14].

Dado que en la mayoría de las señales las frecuencias bajas se prolongan en el tiempo a lo largo de todo el dominio y las frecuencias altas suelen ser eventos puntuales, interesa caracterizar de una forma correcta la frecuencia de las señales de baja frecuencia y tener una buena resolución para las señales de alta frecuencia. Precisamente para cumplir ambas especificaciones está diseñada la transformada Wavelet, ya que su entorno de trabajo es el requerido.

El algoritmo a seguir en la transformada Wavelet es el de utilizar varios filtros paso alto y paso bajo, de modo que en primer lugar se divida el número de datos en dos; una mitad que comprenda la frecuencias altas y la otra que comprenda las frecuencias bajas.

Una vez hecho esto se seguirán subdividiendo las frecuencias bajas hasta llegar a un número prefijado de iteraciones de modo que en los niveles más bajos, en los que se encuentran las frecuencias más bajas, se tendrá un pequeño número de muestras con lo que la resolución temporal será mala y un pequeño abanico de frecuencias, lo que conlleva que la resolución en frecuencia será mucho mejor que la temporal.

Este proceder se conoce como análisis multi-resolución, basado en la transformada Wavelet. Sucederá exactamente la situación opuesta con los niveles más altos de frecuencia, en los que la resolución en frecuencia será mala y la resolución temporal será excelente.

Resumiendo, la transformada Wavelet se diseña de manera que las frecuencias altas tengan mejor resolución en el tiempo porque se dispone de un mayor número de muestras para un mismo espacio de tiempo; por otro lado, las frecuencias más bajas se intentan caracterizar correctamente en frecuencia aun a costa de perder información temporal.

La Teoría de Wavelets tiene muchas aplicaciones reales que comprenden la detección de discontinuidades y puntos de ruptura en las señales, la identificación de frecuencias puras, la reducción de ruido en señales, la compresión de señales, aproximación de funciones, métodos espectrales para resolver ecuaciones diferenciales, análisis de fluidos turbulentos, etc.

Últimamente está cobrando una creciente importancia la conocida como transformada Wavelet packets, que sigue el mismo algoritmo pero se conserva no sólo la información de los filtros paso bajo sino también la de los filtros paso alto, de modo que el zoom se llevará a cabo para la totalidad del espectro de frecuencias.

2.3.5.1 Wavelets

El análisis Wavelet es una herramienta matemática que descompone una señal temporal en suma de diferentes señales temporales denominadas funciones wavelets hijas. Cada una de estas tiene diferentes escalas en diferentes niveles de resolución obtenidos mediante escalado y dilatación de una determinada función matemática temporal denominada función wavelet madre.

Las Wavelets son familias de funciones definidas por:

$$h_{a,b} = \frac{h\left(\frac{x-b}{a}\right)}{\sqrt{|a|}}; a, b \in R, a \neq 0$$

Son generadas a partir de funciones madre h(x). A esa función madre se le agregan coeficientes que son la escala (a) que permite hacer dilataciones y contracciones de la señal y la variable de traslación (b), que permite mover a la señal en el tiempo. Estas variables son números reales y obviamente para una escala de 0 la función queda indeterminada.

Existen diferentes wavelets, utilizadas de forma habitual, que tienen definiciones establecidas. Sin embargo, la elección de un tipo de wavelet depende de la aplicación específica que se le vaya a dar.

Las Wavelets madre más características se muestran a continuación. Véase *Figura 16*.

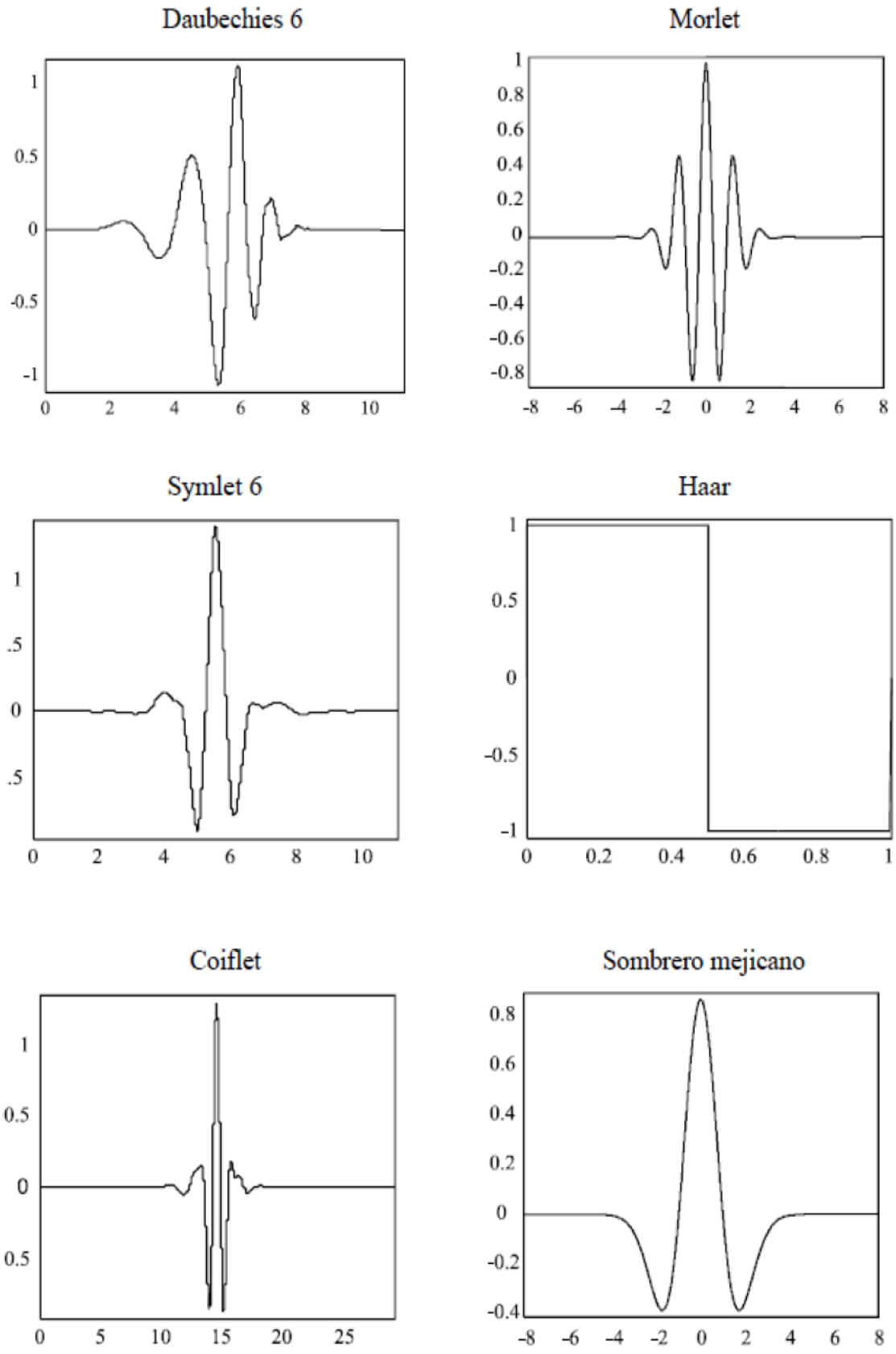


Figura 16. Wavelets madre más usuales

2.3.5.2 *Transformadas de Wavelets*

Las transformadas de Wavelets comprenden la transformada continua de Wavelets y la transformada discreta de Wavelets.

Estas son las herramientas matemáticas que permiten el análisis de señales de manera muy similar que lo hacen las transformadas de Fourier dando información en el dominio del tiempo y en el dominio de la frecuencia.

Transformadas de Wavelets Continuas (CWT)

Las transformadas Wavelets Continuas vienen dadas por la expresión:

$$CWT(a,b) = \sqrt{\left|\frac{f}{f_0}\right|} \int h\left(\frac{f}{f_0}(x-b)\right) f(x)dx; a,b \in R, a \neq 0$$

Para hacer el análisis de una señal se multiplica cada punto de dicha señal por la Wavelet que se haya elegido, cuyas características de escala y traslación serán permanentes para todo el proceso.

Una vez hecho esto cada una de las muestras resultantes se van a sumar y de este modo se obtendrá la señal traslada del dominio del tiempo al dominio de la frecuencia y el tiempo. Este proceso es el mismo que utiliza la transformada de periodo corto de Fourier (STFT).

Transformada Wavelet Discreta (DWT)

Para el caso de la transformación discreta se ha de tener en cuenta un muestreo que convierta la señal continua en discreta. Se define como:

$$d_{j,k} = a_0^{\frac{j}{2}} \int f(x)h(a_0^{-j}x - kb_0)dx$$

El muestreo que se utiliza está basado en el análisis de multiresolución y se realiza en base una serie de filtros paso alto y filtros paso bajo.

De este modo se van obteniendo las muestras de bajas y altas frecuencias. Para esta labor se han diseñado un par de términos importantes que son el decimado y undecimado que propiamente se refieren al sentido en el que se realiza el muestreo.

- **Decimado**: se refiere a incrementar el número de muestras que se toman.
- **Undecimado**: se refiere a decrementar el número de dichas muestras.

Máximos de energía de las Wavelets

El concepto de energía empleado en el análisis wavelet de paquetes está estrechamente ligado a las conocidas nociones derivadas de la teoría de Fourier.

Como paso previo al cálculo de energía se debe seleccionar la wavelet madre Y(t) y el nivel de descomposición N adecuados. La energía en los diferentes niveles de descomposición, desde 1 hasta N, es la energía de los coeficientes dj,k .

La energía de los coeficientes de escala ck empleados para la reconstrucción está definida como la energía del nivel de descomposición N+1. De esta manera la energía para cada nivel de descomposición se define como:

$$E_j = \sum_k \left| d_{j,k} \right|^2 ; j = 1,..., N$$

$$E_{N+1} = \sum_k \left| c_k \right|^2$$

Para obtener la energía relativa se ha de calcular el total de la energía de la señal antes de la descomposición wavelet mediante:

$$E_{total} = \sum_{j=1}^{N+1} E_j$$

Finalmente, la energía relativa está definida como:

$$\rho_j = \frac{E_j}{E_{total}} ; j = 1,..., N+1$$

$$\text{donde } \sum_j \rho_j = 1$$

2.4 Rodamientos y ejes

El presente capítulo desarrolla el método para conseguir las señales de vibración de rodamientos para su posterior estudio. En primer lugar se dará una descripción de los tipos de rodamientos y los defectos más comunes que suelen aparecer en los mismos.

A continuación, se expondrá una introducción teórica acerca de los ejes y su correspondiente mantenimiento.

2.4.1 Rodamientos

Un rodamiento es un elemento mecánico que reduce la fricción entre un eje y las piezas conectadas a éste, que le sirve de apoyo y facilita su desplazamiento. El rodamiento es un elemento normalizado que consta de dos aros concéntricos entre los que se desplazan unos cuerpos rodantes. Estos cuerpos rodantes suelen ir sujetos en la jaula. A continuación se muestran los principales elementos que componen un rodamiento [14]. Véase *Figura 17*.

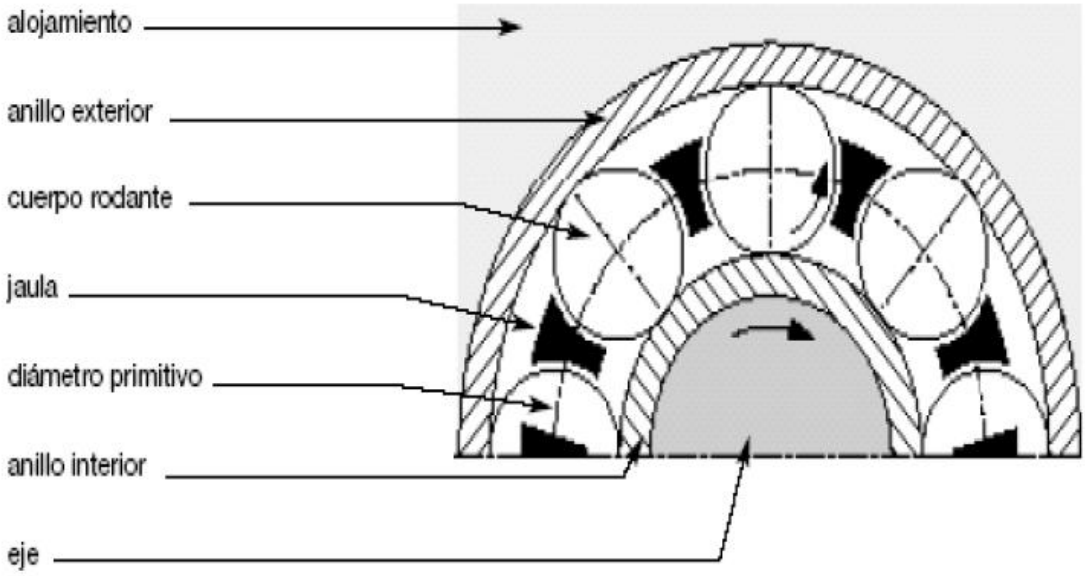


Figura 17. Elementos que componen un rodamiento

2.4.1.1 Tipos de rodamientos

Los tipos de rodamientos más usuales que se pueden encontrar en el mercado bajo los criterios del elemento rodante y su forma de trabajo son los que se muestran en el siguiente diagrama. Véase *Figura 18*.

Figura 18. Tipos de rodamientos

2.4.1.2 Tipos de defectos y causas

Los tipos más significativos de fallos que se pueden dar en rodamientos son:

- **Desgaste:** La presencia de partículas abrasivas (pequeñas indentaciones alrededor de las pistas y las bolas), problemas de lubricación (superficie de contacto brillante, con decoloración café azulada en su última fase) y de vibración (marcas en las pistas, de tipo rectangular para rodamientos de rodillos y circulares en los rodamientos de bolas) pueden provocar un desgaste prematuro del rodamiento que se esté estudiando. Véase *Figura 19*.

Figura 19. Rodamiento con desgaste debido a vibraciones mecánicas

- **Indentación:** Se produce en las pistas y/o elementos rodantes cuando el montaje se realiza con elevada fuerza aplicada de manera brusca o seca (por ejemplo, a martillazos). Estos defectos se transmiten hacia los elementos rodantes pudiéndose producir incrementos de presión considerables en las zonas de contacto elemento rodante - pista. Las partículas abrasivas también pueden causar indentación. Véase *Figura 20*.

Figura 20. Indentación

- **Descascarillado:** Es una consecuencia directa del fenómeno de la fatiga que afecta a los rodamientos. En un determinado momento estas tensiones cíclicas originan una micro fisura que, posteriormente, se va propagando gradualmente con el transcurrir del número de ciclos hacia el exterior de la superficie de los mismos. Los elementos rodantes entran en contacto con este tipo de defecto cíclicamente y contribuyen a que la fisura sea cada vez mayor. Véase *Figura 21*.

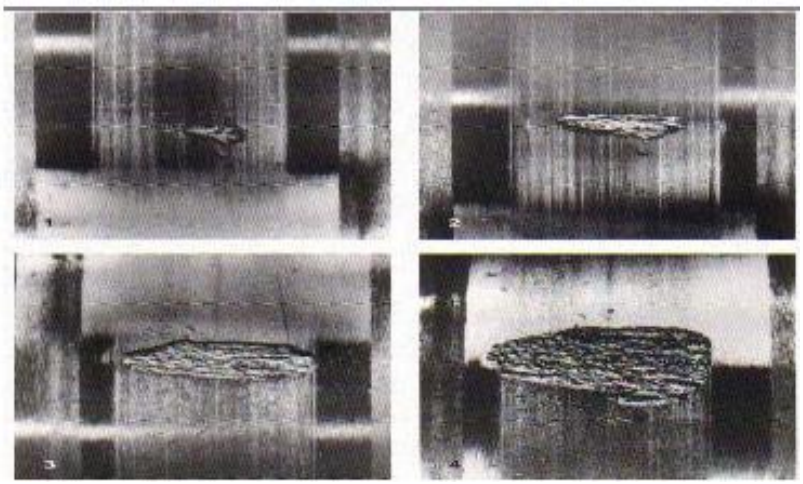


Figura 21. Descascarillado en rodamientos

- **Smearing:** Se produce cuando dos superficies lubricadas incorrectamente deslizan una sobre otra en condiciones de elevada carga. En este caso se originan micro-soldaduras conllevando transferencia de material. cuando se da este fenómeno se alcanzan elevadas temperaturas que provocan concentraciones de tensión que pueden dar lugar a grietas o al desconchado del material del rodamiento. Para evitar la aparición de este fenómeno deben utilizarse métodos adecuados de lubricación, para asegurar la existencia de una capa de lubricante continua y extendida sobre toda la pista de rodadura. Véase *Figura 22*.

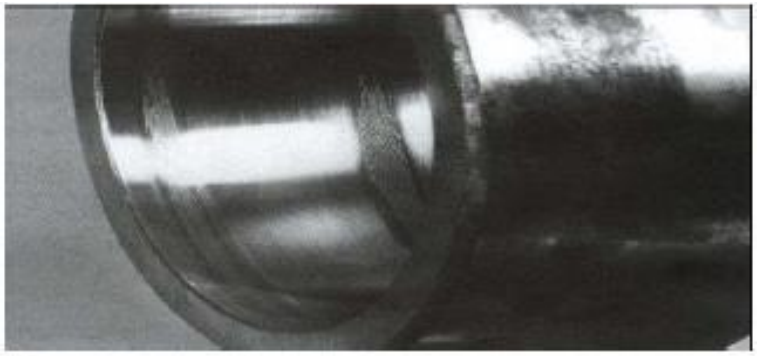


Figura 22. Smearing en un rodamiento

- **Superficies deformadas:** Cuando la película de lubricante entre las pistas y los elementos rodantes es demasiado delgada, los salientes de las superficies rugosas estarán en contacto. a causa de esta interacción se producirán pequeñas grietas en las superficies de rodadura, que en ningún caso deben ser confundidas con la grieta que origina la fatiga clásica del material. estas grietas que originan las superficies deformadas son, en general, de pequeño tamaño y superficiales, pudiendo llegar a ocasionar la incorrecta rodadura de las bolas. Si la lubricación del rodamiento es adecuada no se debe de dar la aparición de los defectos aquí explicados. Véase *Figura 23*.

Figura 23. Rodamiento con la jaula deformada

A continuación se adjunta la representación de los porcentajes pertenecientes a cada una de las causas de los defectos en rodamientos. Véase *Figura 24*.

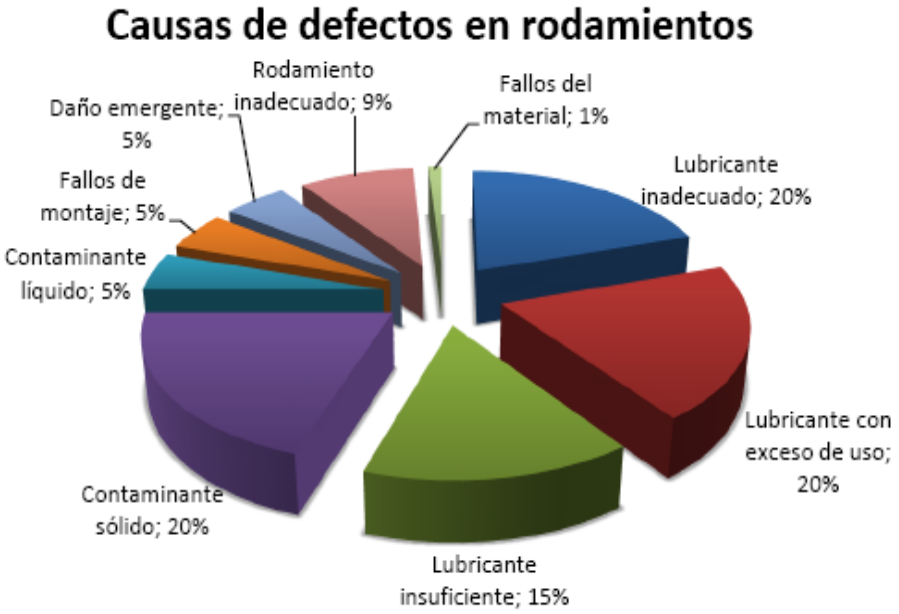


Figura 24. Causas de defectos en rodamientos y porcentajes

Se puede deducir que la principal causa de defectos en rodamientos está íntimamente relacionada con el lubricante aplicado a los mismos, ya sea empleado en exceso como en defecto, como emplear un lubricante inadecuado. Es por ello por lo que se ha llegar a un compromiso entre los dos extremos, a la vez que se selecciona un producto adecuado, para así garantizar el correcto funcionamiento y el cumplimiento de la vida nominal indicada por el fabricante de los mismos.

2.4.2 Ejes

Un eje es un componente fundamental en máquinas para la transmisión de potencia, así como para soportar otros elementos de la máquina. A este elemento, además, se le exige un funcionamiento fiable a unas velocidades y con cargas cada vez mayores. Por todo ello, es conveniente realizar un seguimiento del estado del eje como medida de precaución ante un posible fallo del eje que podría provocar unos serios daños en el resto de la máquina. Véase *Figura 25*.

El principal problema de este método no es el de medir dichas vibraciones, ya que para esto basta con un simple acelerómetro, sino el hecho de distinguir el origen de las mismas, puesto que las vibraciones pueden estar motivadas por diferentes causas como pueden ser desequilibrios, desalineamiento de ejes, holguras de tornillos y tuercas, rozamiento, defectos en rodamientos, fisuras en el eje, etc.

En el caso que nos incumbe, las vibraciones medidas son la entrada al sistema de monitorización y después se procesan con el fin de buscar información relevante del estado de los componentes en estudio. Posteriormente, un sistema clasificador se encarga de proporcionar el diagnóstico de la condición del sistema, lo que disminuye la influencia humana en los resultados del proceso de monitorización.

El objetivo de este método es evaluar el estado dinámico del sistema de manera permanente, ofreciendo un diagnóstico continuado de su condición, lo cual proporciona la ventaja de detectar un cambio en las condiciones de los componentes, es decir, la evolución de un estado "sin defecto" a otro "con defecto", con suficiente antelación.

Figura 25. Eje de rodadura lineal

3. Programación y adquisición de datos

3.1 Interfase Hombre Máquina

La sigla HMI es la abreviación en inglés de Interfaz Hombre Máquina. Los sistemas HMI podemos pensarlos como una ventana de un proceso. Esta ventana puede estar en dispositivos especiales como paneles de operador o en una computadora. Los sistemas HMI en computadoras se los conoce también como software HMI o de monitoreo y control de supervisión. Las señales del proceso son conducidas al HMI por medio de dispositivos como tarjetas de entrada/salida en la computadora, PLC's (Controladores lógicos programables), RTU (Unidades remotas de I/O) o DRIVE's (Variadores de velocidad de motores). Todos estos dispositivos deben tener una comunicación que entienda el HMI [16]. Véase *Figura 26*.

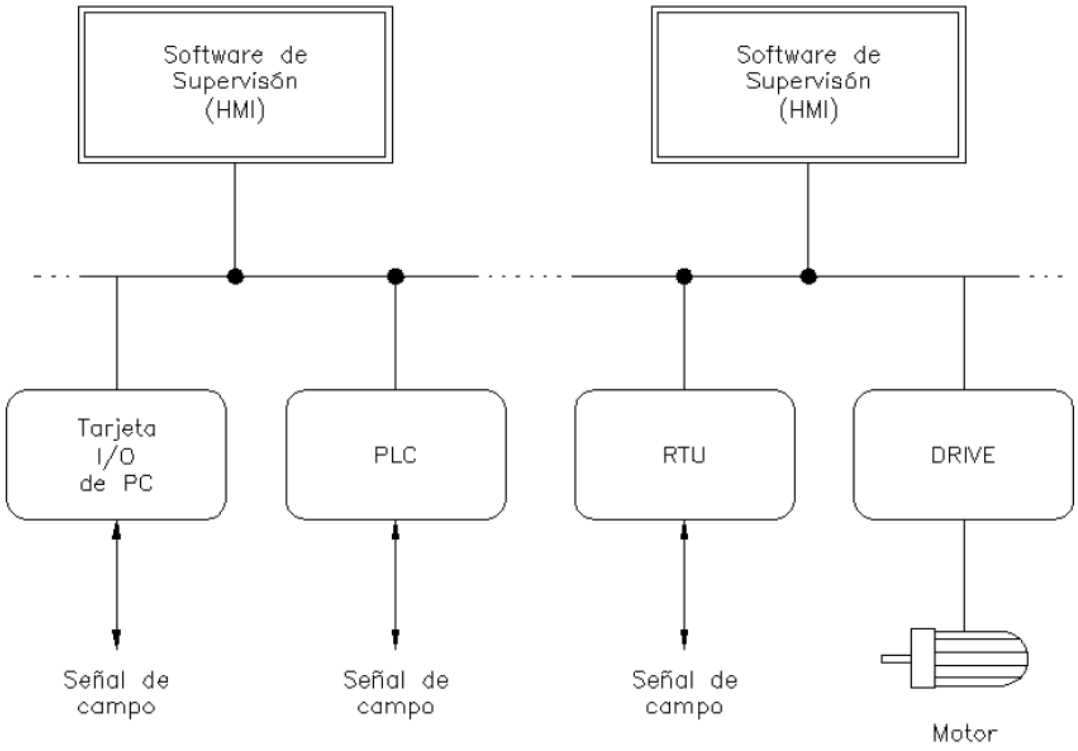


Figura 26. Sistema HMI

La estructura de los software's HMI está compuesta por un conjunto de programas y archivos. Hay programas para diseño y configuración del sistema y otros que son el motor mismo del sistema. En la *Figura 27* se muestra cómo funcionan algunos de los programas y archivos más importantes. Los rectángulos de la figura representan programas y las elipses representan archivos. Los programas que están con recuadro simple representan programas de diseño o configuración del sistema. Los que tienen doble recuadro representan programas que son el motor del HMI.

Con los programas de diseño, como el "Editor de pantallas" se crea moldes de pantallas para visualización de datos del proceso. Estos moldes son guardados en archivos "Archivo de pantalla" y almacenan la forma como serán visualizados los datos en las pantallas.

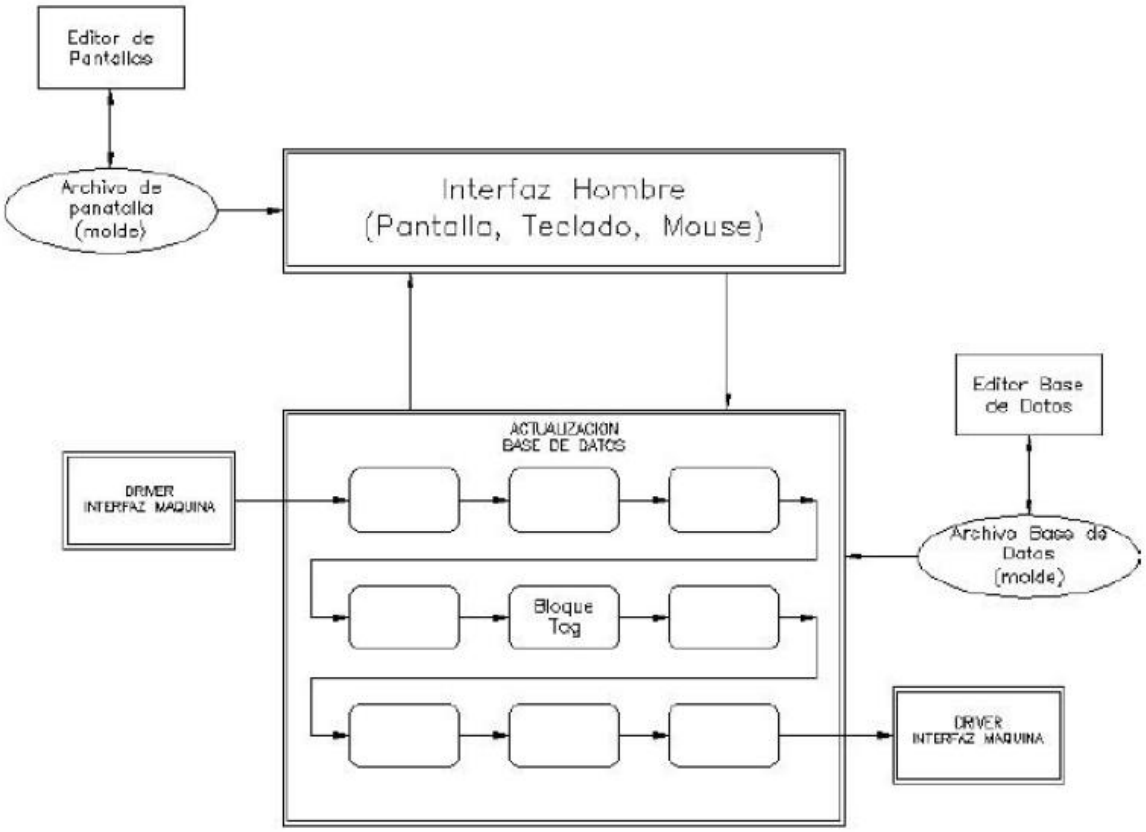


Figura 27. Estructura general del software HMI

Algunas de las funciones de un software HMI son:

- Monitoreo. Es la habilidad de obtener y mostrar datos de la planta en tiempo real. Estos datos se pueden mostrar como números, texto o gráficos que permitan una lectura más fácil de interpretar.

- Supervisión. Esta función permite junto con el monitoreo la posibilidad de ajustar las condiciones de trabajo del proceso directamente desde la computadora.

- Alarmas. Es la capacidad de reconocer eventos excepcionales dentro del proceso y reportarlo estos eventos. Las alarmas son reportadas basadas en límites de control preestablecidos.

- Control. Es la capacidad de aplicar algoritmos que ajustan los valores del proceso y así mantener estos valores dentro de ciertos límites. Control va más allá del control de supervisión removiendo la necesidad de la interacción humana. Sin embargo la aplicación de esta función desde un software corriendo en una PC puede quedar limitada por la confiabilidad que quiera obtenerse del sistema.

- Históricos. Es la capacidad de mostrar y almacenar en archivos, datos del proceso a una determinada frecuencia. Este almacenamiento de datos es una poderosa herramienta para la optimización y corrección de procesos.

Algunas de las tareas de un software HMI de supervisión y control son:

- Permitir una comunicación con dispositivos de campo.

- Actualizar una base de datos dinámica con las variables del proceso.

- Visualizar las variables mediante pantallas con objetos animados (mímicos).

- Permitir que el operador pueda enviar señales al proceso, mediante botones, controles ON/OFF, ajustes continuos con el mouse o teclado.

- Supervisar niveles de alarma y alertar/actuar en caso de que las variables excedan los límites normales.

- Almacenar los valores de las variables para análisis estadístico y/o control.

- Controlar en forma limitada ciertas variables de proceso.

3.2 Interfaces gráficas y entornos de programación gráfica

Con la idea de simplificar el uso de los ordenadores para usuarios de todo tipo y no sólo para los expertos, se ha convertido en una práctica habitual utilizar metáforas visuales por medio de la llamada interfaz gráfica de usuario (IGU ó GUI en inglés) para que el usuario interactúe y establezca un contacto más fácil e intuitivo con el ordenador. En estos casos, un simple clic de ratón sobre algún gráfico que aparece en la pantalla, sustituye a la tediosa tarea de escribir código fuente para que el ordenador interprete que debe realizar alguna acción. En 1981 aparecieron los primeros ordenadores personales, los llamados Pc's, pero hasta 1993 no se generalizaron las interfaces gráficas de usuario. El escritorio del sistema operativo Windows de Microsoft y su sistema de ventanas sobre la pantalla se ha estandarizado y universalizado, pero fueron los ordenadores Macintosh de la compañía Apple los primeros que introdujeron las interfaces gráficas de usuario [16].

Una interfaz es un dispositivo que permite comunicar dos sistemas que no hablan el mismo lenguaje. Restringido a aspectos técnicos, se emplea el término interfaz para definir el juego de conexiones y dispositivos que hacen posible la comunicación entre dos sistemas. Sin embargo, cuando aquí se habla de interfaz se refiere a la cara visible de los programas tal y como se presenta a los usuarios para que interactúen con la máquina. La interfaz gráfica implica la presencia de un monitor de ordenador o pantalla constituida por una serie de menús e iconos que representan las opciones que el usuario puede tomar dentro del sistema.

La interfaz proporcionará al usuario el conjunto de posibilidades que podrá seguir durante todo el tiempo que se relacione con el programa, detallando lo que verá y escuchará en cada momento, y las acciones que puede realizar, así como las respuestas que puede ofrecer el sistema. El usuario, además de entender el mensaje, ha de comprender la mecánica operativa que se le ofrece (sintaxis, órdenes, códigos, abreviaturas, iconos, etc.). Una buena interfaz requiere poco esfuerzo por parte del usuario, simplicidad y funcionalidad.

Las características básicas de una buena interfaz podrían sintetizarse en:

- Facilidad de comprensión, aprendizaje y uso.
- Representación fija y permanente de un determinado contexto de acción (fondo).
- El objeto de interés ha de ser de fácil identificación.
- Diseño ergonómico mediante el establecimiento de menús, barras de acciones e iconos de fácil acceso.
- Las interacciones se basarán en acciones físicas sobre elementos de código visual o auditivo (iconos, botones, imágenes, mensajes de texto o sonoros, barras de desplazamiento y navegación...) y en selecciones de tipo menú con sintaxis y órdenes.

- Las operaciones serán rápidas, incrementales y reversibles, con efectos inmediatos.
- Existencia de herramientas de Ayuda y Consulta.
- Tratamiento del error bien cuidado y adecuado al nivel de usuario.

La tipografía y el tratamiento del color son dos elementos a los que hay que prestar especial importancia a la hora de establecer una buena interfaz, poniendo especial cuidado en el diseño de las formas y la coherencia interna entre ellas.

Para diseñar una buena interfaz enfocada hacia el usuario es necesario tener claros los objetivos del hipertexto, teniendo en cuenta no sólo lo que se persigue ofreciendo información, sino las necesidades que van a tener los usuarios a la hora de consultarlo. También es clave determinar el contenido y la funcionalidad, especificar la estructura organizativa, la navegación, las secciones y los sistemas de búsqueda. Hay que tener en cuenta que cada usuario puede tener diferentes necesidades y un buen sistema de navegación debe contar con las herramientas adecuadas para diferentes funciones. Como cada usuario pude tener diferentes necesidades, es importante ofrecer diferentes formas de acceso y búsqueda, desde búsquedas precisas, hasta exploraciones guiadas o a elección del lector.

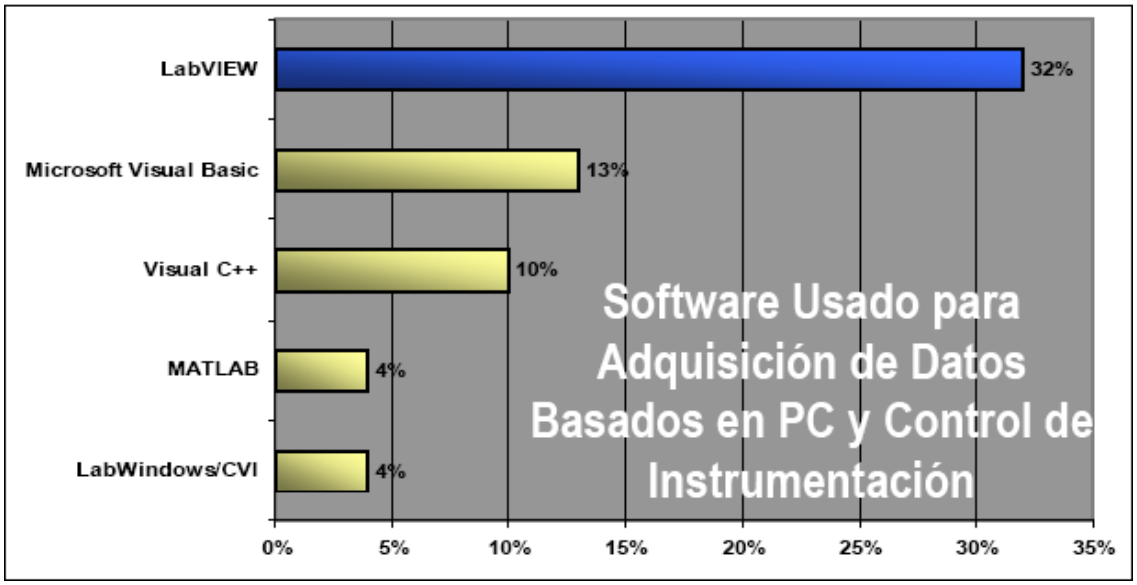


Figura 28. Comparación de Software para el desarrollo de HMI's

Los lenguajes de programación visual como Visual C++, Visual Basic, LabVIEW, MATLAB, etc., son entornos de programación gráfica y se utilizan para desarrollar software HMI a medida del usuario, es decir, interfaces gráficas de usuario. Véase *Figura 28*.

A continuación, se exponen algunos de los más importantes entornos de programación gráfica utilizados para la creación de GUI's.

3.2.1 MATLAB (GUIDE)

Para el desarrollo de la aplicación en tiempo real, se trabaja con una herramienta de MATLAB llamada GUIDE (Graphical Use Interface Development Environment). Esta herramienta está pensada para desarrollar GUI's (Graphical User Interfaces) fácil y rápidamente haciendo sencillo el diseño y presentación de los controles de la interfaz, reduciendo la labor en el momento de seleccionar, deshacer, arrastrar y centrar controles, así como la personalización de las propiedades de estos.

El proceso a seguir para el desarrollo de un programa mediante GUIDE es que una vez se tienen todos los controles en posición, se editan las funciones de llamada (Callback) de cada uno de ellos, escribiendo el código de MATLAB que se ejecutará cuando el control sea utilizado. GUIDE está diseñado para hacer menos tedioso el proceso de desarrollo de la interfaz gráfica, para ello cuenta con un editor de propiedades (property editor) con el que se podrá modificar en cualquier momento los nombres, valores por defecto y las propiedades de los elementos.

Encontraremos doce herramientas con las que podemos crear nuestra interfaz: Push Button, Toggle Button, Radio Button, CheckBox, Edit Text, Static Text, Slider, Table, Panel, ListBox, Popup Menu, Axes. Así mismo, encontraremos un formulario en blanco sobre el cual podremos empezar a construir el nuestro proyecto.

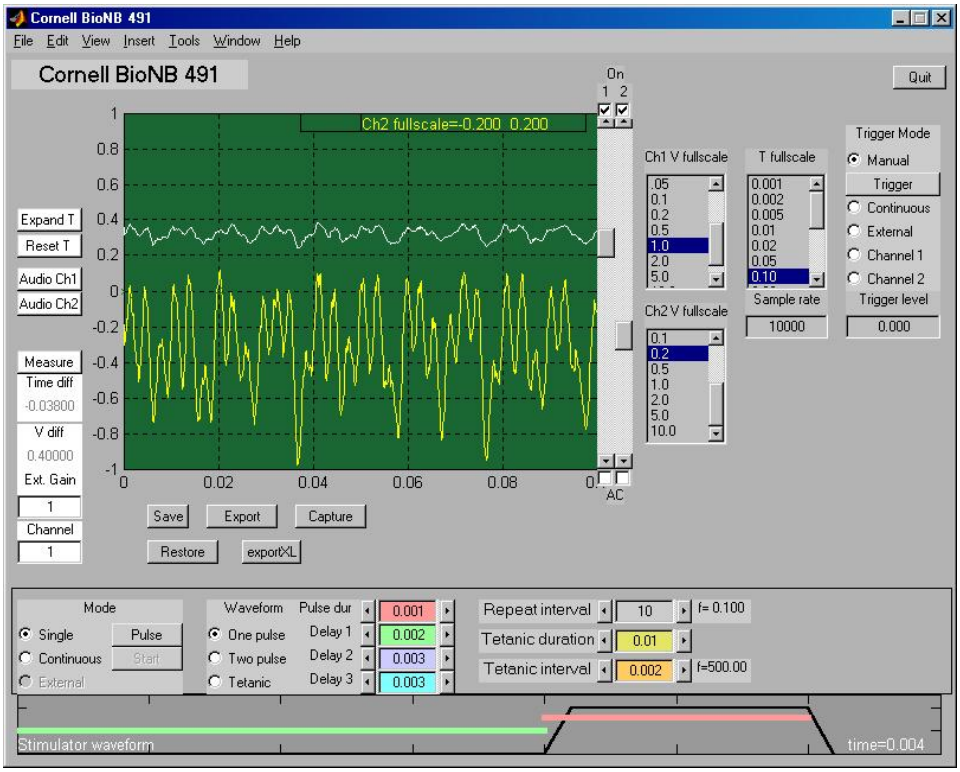


Figura 29. Interfaz gráfica en MATLAB

Gracias a toda esta cantidad de múltiples propiedades gráficas, el usuario podrá realizar un diseño de GUI adaptado a sus necesidades y preocuparse sólo por la programación del software. Véase *Figura 29*.

3.2.2 LabVIEW

Es una herramienta gráfica de programación, esto significa que los programas no se escriben, sino que se dibujan, facilitando su comprensión. Al tener ya pre-diseñados una gran cantidad de bloques, se le facilita al usuario la creación del proyecto, con lo cual en vez de estar una gran cantidad de tiempo en programar un dispositivo/bloque, se le permite invertir mucho menos tiempo y dedicarse un poco más en la interfaz gráfica y la interactuación con el usuario final. Cada hoja de trabajo consta de dos partes diferenciadas:

- **Panel Frontal**. Es la interfaz con el usuario, se utiliza para interactuar con el usuario cuando el programa se esta ejecutando. Los usuarios podrán observar los datos del programa actualizados en tiempo real y como van fluyendo los datos de entradas y salidas. En esta interfaz se definen los controles que se usan como entradas, pueden ser botones, marcadores, etc. También se definen los indicadores que se usan como salidas, pueden ser gráficas, matrices, etc.

- **Diagrama de Bloques**. Es el programa propiamente dicho, donde se define su funcionalidad, aquí se colocan iconos que realizan una determinada función y se interconectan con el código que controla el programa. Suele haber una tercera parte icono/conector que son los medios utilizados para conectar una hoja de trabajo con otras hojas de trabajo. Véase *Figura 30*.

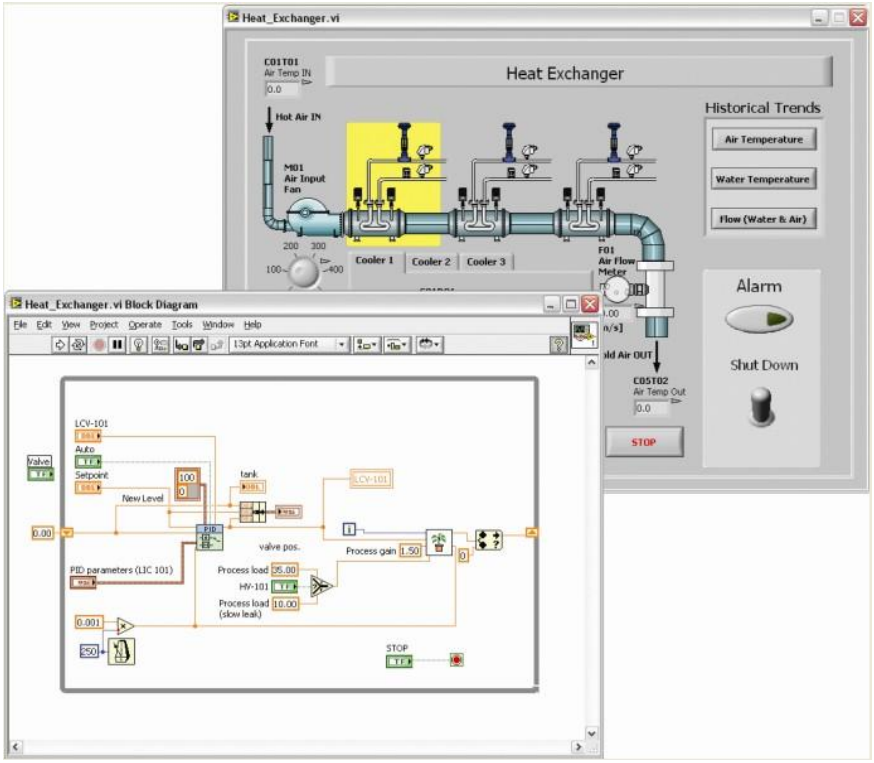


Figura 30. Interfaz gráfica en LabVIEW

3.2.3 Delphi

Delphi es un entorno de desarrollo de software diseñado para la programación de propósito general con énfasis en la programación visual.
Un uso habitual de Delphi, aunque no el único, es el desarrollo de aplicaciones visuales y de bases de datos cliente-servidor y multicapas. Debido a que es una herramienta de propósito múltiple, se usa también para proyectos de casi cualquier tipo, incluyendo aplicaciones de consola, aplicaciones de web, servicios COM y DCOM y servicios del sistema operativo.

Como entorno visual, la programación en Delphi consiste en diseñar los formularios que componen al programa colocando todos sus controles (botones, etiquetas, campos de texto, etc.) en las posiciones deseadas, normalmente usando un ratón. Luego se asocia código a los eventos de dichos controles y también se pueden crear módulos de datos, que regularmente contienen los componentes de acceso a datos y las reglas de negocio de una aplicación.

Una de las principales características y ventajas de Delphi es su capacidad para desarrollar aplicaciones con conectividad a bases de datos de diferentes fabricantes. El programador de Delphi cuenta con una gran cantidad de componentes para realizar la conexión, manipulación, presentación y captura de los datos, algunos de ellos liberados bajo licencias de código abierto o gratuito.
Estos componentes de acceso a datos pueden enlazarse a una gran variedad de controles visuales, aprovechando las características del lenguaje orientado a objetos, gracias al polimorfismo.

Una gran parte de los componentes disponibles para Delphi son controles que encapsulan los elementos de interacción con el usuario como botones, menús, barras de desplazamiento, etc.

Además de poder utilizar en un programa estos componentes estándar (botones, grillas, conjuntos de datos, etc.), es posible crear nuevos componentes o mejorar los ya existentes, extendiendo la funcionalidad de la herramienta. En internet existe un gran número de componentes, tanto gratuitos como comerciales, disponibles para los proyectos a los que no les basten los que vienen ya con la herramienta.

3.2.4 Visual Basic

Visual Basic es un lenguaje de programación que fue desarrollado con la intención de simplificar la programación utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y en cierta medida también la programación misma.

Visual Basic constituye un IDE (entorno de desarrollo integrado o en inglés Integrated Development Enviroment) que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código (programa donde se escribe el código fuente), un depurador (programa que corrige errores en el código fuente para que pueda ser bien compilado), un compilador (programa que traduce el código fuente a lenguaje de máquina) y un constructor de interfaz gráfica o GUI (es una forma de programar en la que no es necesario escribir el código para la parte gráfica del programa, sino que se puede hacer de forma visual).

La ventana de propiedades contiene diferentes formas para utilizar el programa, las cuales son: (Pointer) Apuntador o puntero, (Label) Etiqueta, (Frame) Marco, (CheckBox) Casilla de verificación, (ComboBox) Lista desplegable, (HScrollBar) Barra de desplazamiento horizontal, (Timer) Temporizador, (DirListBox) Lista de directorios, (Shape) Figura, (Image) Imagen, (PictureBox) Caja de Imagen, (TextBox) Caja de texto, (CommandButton) Botón de pulsación, (OptionButton) Botón de opción, (ListBox) Lista, (VScrollBar) Barra de desplazamiento vertical, (DriveListBox) Lista de unidades de disco, (FileListBox) Lista de archivos, (Line) Línea y por último (Data) Datos.

Gracias a toda esta cantidad de múltiples propiedades gráficas, el usuario podrá realizar un diseño de GUI adaptado a sus necesidades y preocuparse sólo por la programación del software.

3.3 Entorno de programación MATLAB

3.3.1 Entorno gráfico y creación de interfaces

Hoy en día, con la proliferación de los sistemas operativos tipo Windows, en los que el usuario encuentra un ambiente amigable y fácil de usar, donde se puede ingresar datos en cajas de texto y ejecutar códigos con sólo hacer clic en un botón, MATLAB no podía quedarse atrás [16].

MATLAB es un gran programa de cálculo técnico y científico. Para ciertas operaciones es muy rápido, cuando puede ejecutar sus funciones en código nativo con los tamaños más adecuados para aprovechar sus capacidades de vectorización. En otras aplicaciones resulta bastante más lento que el código equivalente desarrollado en C/C++ o Fortran. Sin embargo, siempre es una magnífica herramienta de alto nivel para desarrollar aplicaciones técnicas, fácil de utilizar y que aumente significativamente la productividad de los programadores respecto a otros entornos de desarrollo.

Las últimas versiones de MATLAB se han preocupado por incorporar herramientas fáciles de usar, con las que el usuario pueda crear interfaces gráficas para la interacción con sus programas.

MATLAB posee la herramienta GUIDE (Graphical Use Interface Development Environment) que permiten crear interfaces gráficas y tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++ pero con muchas menos posibilidades. Estas son las llamadas GUI (Guide User Interface) y podemos acceder a ellas tecleando la instrucción “guide” en la ventana de comandos.

Encontraremos doce herramientas con las que podemos crear nuestra interfaz: Push Button, Toggle Button, Radio Button, CheckBox, Edit Text, Static Text, Slider, Table, Panel, ListBox, Popup Menu, Axes. Así mismo, encontraremos un formulario en blanco sobre el cual podemos empezar a construir el nuestro proyecto.

En general, familiarizarse a la programación de una interfaz gráfica de MATLAB es un poco más complicado que programarla en otros programas de creación de interfaces gráficas. Por ello se puede usar el comando “Help”, que en sus últimas versiones tiene guías paso a paso para crear GUI’s.

Sin embargo, el entorno de diseño y trabajo de MATLAB es muy gráfico, intuitivo y fácil de manejar.

3.3.2 Características y tipos

Algunas de las características principales que hacen del MATLAB un programa intuitivo y de gran utilidad técnica se exponen a continuación:

- Computación numérica para obtener rápidamente resultados precisos.
- Gráficos para visualizar y analizar los datos.
- Lenguaje y entorno de programación interactivos.
- Herramientas para construir GUI's a medida.
- Integración con aplicaciones externas como C, C++, Fortran, Java, componentes COM o Excel.
- Posibilidad de importar datos desde archivos y dispositivos externos y de usar archivos E/S de bajo nivel (además de acceso a bases de datos y hardware adicional a través de productos añadidos).
- Conversión de aplicaciones MATLAB a C y C++ mediante su compilador.

Este amplio conjunto de prestaciones hacen de MATLAB la base ideal para el desarrollo de soluciones a problemas matemáticos, creación de interfaces gráficas y comunicaciones con otros programas.

Dentro de las interfaces de usuario se puede distinguir básicamente tres tipos:

- Una interfaz de hardware, a nivel de los dispositivos utilizados para ingresar, procesar y entregar los datos: teclado, ratón y pantalla visualizadora.
- Una interfaz de software, destinada a entregar información acerca de los procesos y herramientas de control, a través de lo que el usuario observa habitualmente en la pantalla.
- Una interfaz de Software-Hardware, que establece un puente entre la máquina y las personas, permite a la maquina entender la instrucción y a el hombre entender el código binario traducido a información legible.

3.3.3 Entorno flexible de trabajo

El entorno de trabajo de MATLAB es muy gráfico e intuitivo, similar al de otras aplicaciones profesionales de *Windows*.

Está diseñado para la computación interactiva y automatizada. Mediante las funciones matemáticas y gráficas incorporadas y las herramientas de fácil manejo se pueden analizar y visualizar los datos de un vistazo. El lenguaje estructurado y las herramientas de programación permiten guardar los resultados de las exploraciones interactivas y desarrollar algoritmos y aplicaciones.

Los usuarios que trabajan con una amplia gama de aplicaciones de diversos niveles de complejidad han encontrado en MATLAB un entorno eficaz y flexible que evoluciona a su mismo ritmo.

Con MATLAB se puede experimentar de forma interactiva y probar ideas propias con sólo aplicar la gama de funciones de análisis y gráficos del producto. El escritorio de MATLAB permite acceder fácilmente a sus archivos de datos, programas, gráficos, ayuda en línea del producto y mucho más. También puede configurarse para que contenga las herramientas con las que se trabaja más a menudo.

A través de la interfaz del escritorio de MATLAB, se puede personalizar para adaptarlo a cualquier estilo de trabajo y usar selectivamente las funciones que sean necesarias en cada fase del proyecto.

Los componentes más importantes del entorno de trabajo de MATLAB son:

1. El Escritorio de MATLAB (*MATLAB Desktop*)

   El *MATLAB Desktop* es la ventana más general de la aplicación. El resto de las ventanas o componentes pueden alojarse en el *MATLAB Desktop* o ejecutarse como ventanas independientes.

   A su vez, los componentes alojados en el *MATLAB Desktop* pueden aparecer como sub-ventanas independientes o como pestañas dentro de una de las sub-ventanas.

   MATLAB ofrece una gran flexibilidad al respecto y es cada usuario quien decide en qué forma desea utilizar la aplicación.

   La *Figura 31* muestra el menú *Desktop*, desde el que se controlan los componentes visibles y la forma en que se visualizan. La configuración adoptada por el usuario se mantendrá la siguiente vez que arranque el programa.

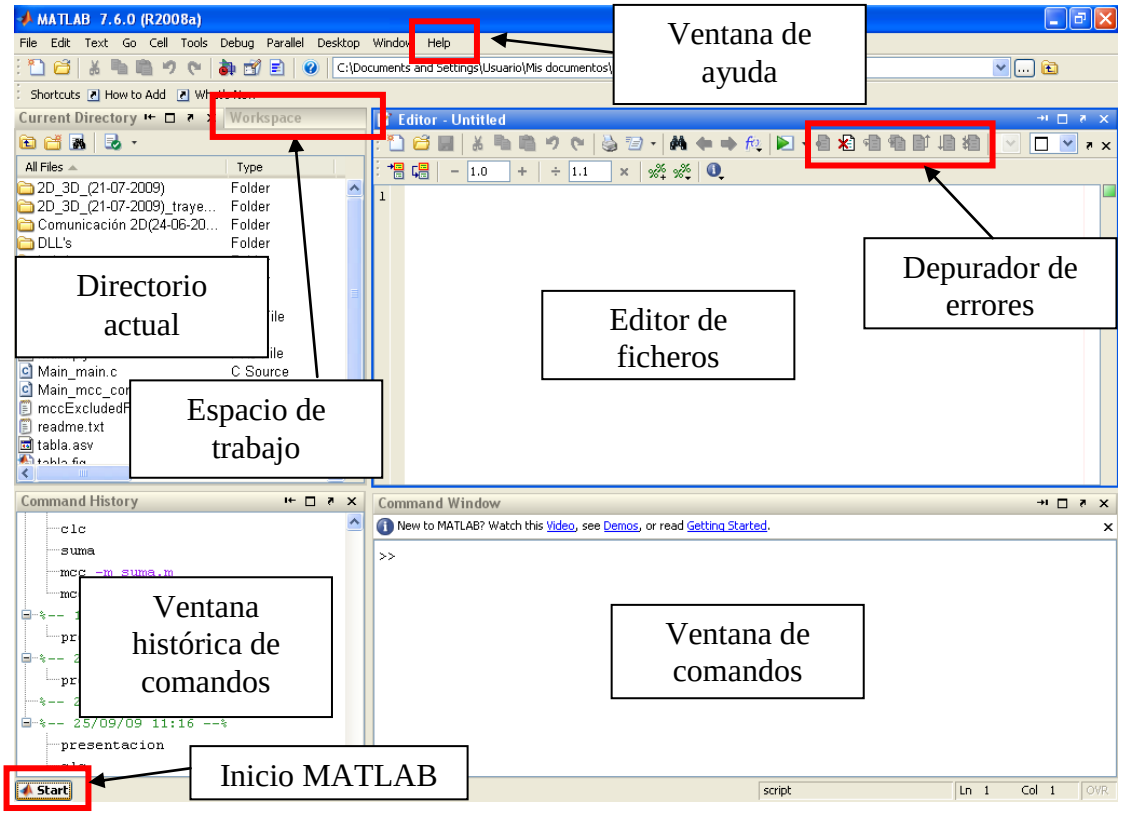


Figura 31. Vista del escritorio de MATLAB (MATLAB Desktop)

2. **Los componentes individuales orientados a tareas concretas**:
   a) La ventana de comandos (*Command Window*).
   b) La ventana histórica de comandos (*Command History*).
   c) El directorio actual (*Current Directory*).
   d) El espacio de trabajo (*Workspace*).
   e) El editor de ficheros y depurador de errores (*Editor&Debugger*).
   f) La ventana de ayuda (*Help*).

En la parte inferior izquierda de la pantalla aparece el botón *Start*, con una función análoga a la del botón *Inicio* de *Windows*. *Start* da acceso inmediato a los módulos o componentes de MATLAB que se tengan instalados.

A continuación se describen brevemente estos componentes. Utilizar MATLAB y desarrollar programas para MATLAB es mucho más fácil si se conoce bien este entorno de trabajo para alcanzar la máxima productividad personal en el uso de esta aplicación.

a) La ventana de comandos (*Command Window*)

Ésta es la ventana en la que se ejecutan interactivamente las instrucciones de MATLAB y en donde se muestran los resultados correspondientes si se ejecutan operaciones matemáticas. Es la ventana más importante y la única que existía en las primeras versiones de la aplicación.

b) La ventana histórica de comandos (*Command History*)

La ventana *Command History* ofrece acceso a las sentencias que se han ejecutado anteriormente en la *Command Window*. Estas sentencias están también accesibles por medio de las teclas ↑ y ↓ como en las versiones anteriores, pero esta ventana facilita mucho el tener una visión más general de lo hecho anteriormente y seleccionar lo que realmente se desea repetir.
Las sentencias ejecutadas anteriormente se pueden volver a ejecutar mediante un doble clic o por medio del menú contextual que se abre al clicar sobre ellas con el botón derecho. También se pueden copiar y volcar sobre la línea de comandos, pero se ha de copiar toda la línea, sin que se admita la copia de un fragmento de la sentencia. Existen opciones para borrar algunas o todas las líneas de esta ventana.

c) El directorio actual (*Current Directory*)

La ventana *Current Directory* permite explorar los directorios del ordenador en forma análoga a la del *Explorador* u otras aplicaciones de *Windows*. Cuando se llega al directorio deseado se muestran los ficheros allí contenidos. La ventana *Current Directory* permite ordenarlos por fecha, tamaño, nombre, etc. El directorio actual cambia automáticamente en función del directorio seleccionado con este explorador, y también se puede cambiar desde la propia barra de herramientas del *Matlab Desktop*.

Para que un fichero **.m* se pueda ejecutar es necesario que se cumpla una de las dos condiciones siguientes:

1. Que esté en el *directorio actual*. Este directorio es el primer sitio en el que MATLAB busca cuando desde la línea de comandos se le pide que ejecute un fichero.

2. Que esté en uno de los directorios indicados en el *Path* de MATLAB. El *Path* es una lista de directorios donde el programa busca los ficheros que ha de ejecutar. Muchos de los directorios del *Path* son propios de MATLAB, pero los usuarios pueden añadir sus propios directorios.

d) El espacio de trabajo (*Workspace*)

El espacio de trabajo de MATLAB (*Workspace*) es el conjunto de variables y de funciones de usuario que en un determinado momento están definidas en la memoria del programa o de la función que se está ejecutando.

Éstas son las variables del espacio de trabajo base, es decir, el de la línea de comandos de MATLAB. Cada función tiene su propio espacio de trabajo, con variables cuyos nombres no interfieren con las variables de los otros espacios de trabajo.

La ventana *Workspace* constituye un entorno gráfico para ver las variables definidas en el espacio de trabajo. La *Figura 31* muestra el aspecto inicial de la ventana *Workspace* cuando se abre desde un determinado programa.

Hay que tener en cuenta que cuando se termina de ejecutar una función y se devuelve el control al programa que la había llamado, las variables definidas en la función dejan de existir salvo que se hayan declarado persistentes y también deja de existir su espacio de trabajo.

e) El editor de ficheros y depurador de errores (*Editor&Debugger*)

En MATLAB tienen particular importancia los ficheros *m-files*. Son ficheros de texto ASCII, con la extensión **.m*, que contienen conjuntos de comandos o definición de funciones. La importancia de estos ficheros **.m* es que al teclear su nombre en la línea de comandos y pulsar *Intro*, se ejecutan uno tras otro todos los comandos contenidos en dicho fichero. El poder guardar instrucciones y grandes matrices en un fichero permite ahorrar mucho trabajo de tecleado.

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

El *Editor* muestra con diferentes colores los diferentes tipos o elementos constitutivos de los comandos, en verde los comentarios, en violeta las cadenas de caracteres, etc. El *Editor* se preocupa también de que las comillas o paréntesis que se abren, no se queden sin el correspondiente elemento de cierre. Seleccionando varias líneas y clicando con el botón derecho aparece un menú contextual cuya sentencia "Comment" permite entre otras cosas comentar con el carácter *%* todas las líneas seleccionadas.

Estos comentarios pueden volver a su condición de código ejecutable seleccionándolos y ejecutando "Uncomment" en el menú contextual.

Otra opción muy útil de ese menú contextual es "Smart Indent", que organiza el sangrado de los bucles y bifurcaciones de las sentencias seleccionadas.

La ejecución del *Debugger* comienza eligiendo el comando *Run* en el menú *Debug*, pulsando la tecla F5, clicando en el botón *Continue* () de la barra de herramientas del *Editor* o tecleando el nombre del fichero en la línea de comandos de la *Command Window*.

Los puntos rojos que aparecen en el margen izquierdo son "breakpoints", es decir, puntos en los que se detiene la ejecución de programa. La flecha verde en el borde izquierdo indica la sentencia en que está detenida la ejecución antes de ejecutar dicha sentencia.

Cuando el cursor se coloca sobre una variable aparece una pequeña ventana con los valores numéricos de esa variable.

A continuación, se detalla el significado de los botones pertenecientes al depurador de errores:

 **Set/Clear Breakpoint**. Coloca o borra un "breakpoint" en la línea en que está el cursor.

 **Clear All Breakpoints**. Elimina todos los "breakpoints" que haya en el fichero.

 **Step**. Avanzar un paso sin entrar en las funciones de usuario llamadas en esa línea.

 **Step In**. Avanzar un paso, y si en ese paso hay una llamada a una función cuyo fichero **.m* está accesible, entra en dicha función.

 **Step Out**. Salir de la función que se está ejecutando en ese momento.

 **Continue**. Continuar la ejecución hasta el siguiente "breakpoint".

 **Quit Debugging**. Terminar la ejecución del *Debugger*.

El *Debugger* es muy útil para detectar y corregir errores y además sirve para aprender métodos numéricos y técnicas de programación.

f) La ventana de ayuda (*Help*)

MATLAB dispone de un excelente menú de ayuda (*Help*) con el que se puede encontrar la información que se desee. La *Figura 32* muestra las distintas opciones que aparecen en el menú *Help* de la ventana principal.

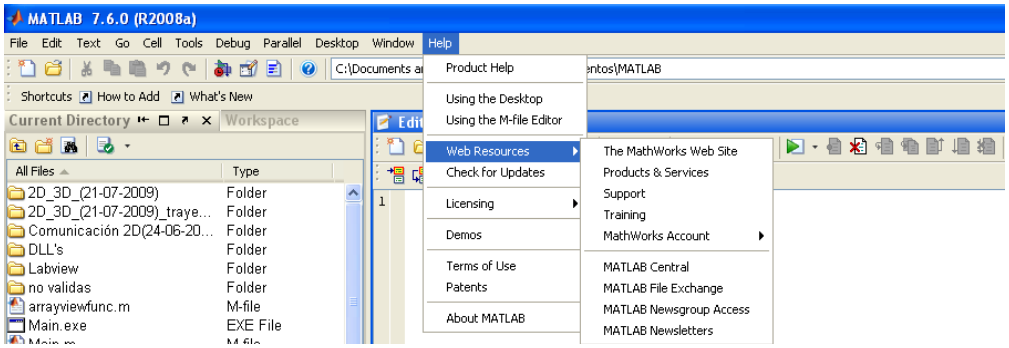


Figura 32. Algunas páginas web sobre MATLAB

En resumen, MATLAB dispone de una ayuda muy completa y accesible, estructurada en varios niveles (línea de comandos en la *Command Window*, ventana *Help*, y manuales en formato PDF), con la que es muy importante estar familiarizado, porque hasta los más expertos programadores tienen que acudir a ella con una cierta frecuencia.

3.3.4 Diseño y configuración de la GUI

Para iniciar la herramienta GUIDE que se va a utilizar a la hora de realizar el proyecto, se pueda hacer de dos maneras:

- Ejecutando la siguiente instrucción en la ventana de comandos:

```
>> guide
```

- Haciendo un clic en el icono que muestra la *Figura 33*

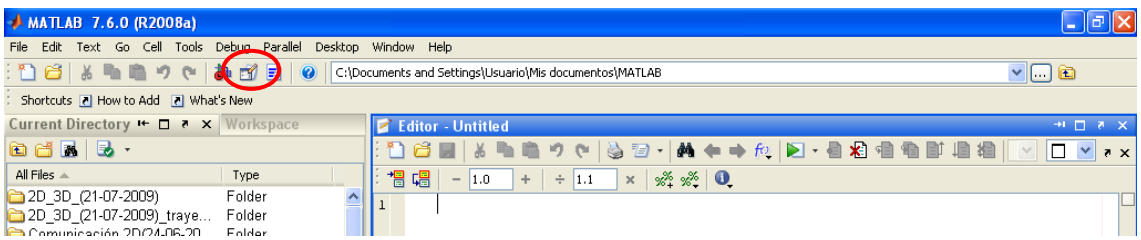


Figura 33. Icono de inicio de la herramienta GUIDE

Se presentan las siguientes opciones:

a) GUI en blanco (Por defecto)

La opción de interfaz gráfica de usuario en blanco (viene predeterminada), nos presenta un formulario nuevo, en el cual podemos diseñar nuestro programa.

b) GUI con Controles

Esta opción presenta un ejemplo en el cual se calcula la masa, dada la densidad y el volumen, en alguno de los dos sistemas de unidades. Podemos ejecutar este ejemplo y obtener resultados.

c) GUI con Ejes y Menú

Esta opción es otro ejemplo el cual contiene el menú *File* con las opciones *Open*, *Print* y *Close*. En el formulario tiene un *Popup menu*, un *push button* y un objeto *Axes*, podemos ejecutar el programa eligiendo alguna de las seis opciones que se encuentran en el menú despegable y haciendo clic en el botón de comando.

d) Pregunta de diálogo Modal

Con esta opción se muestra en la pantalla un cuadro de diálogo común, el cual consta de una pequeña imagen, una etiqueta y dos botones "Yes" y "No", dependiendo del botón que se presione, el GUI retorna el texto seleccionado (la cadena de caracteres 'Yes' o 'No').

Elegimos la primera opción, GUI en blanco. Véase *Figura 34*.

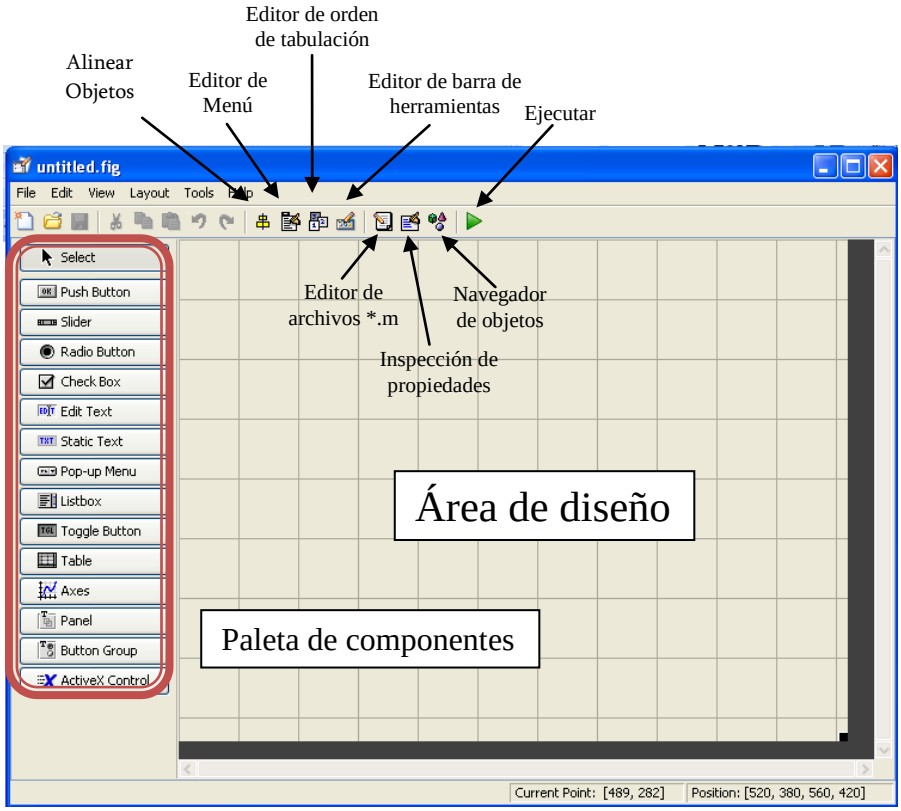


Figura 34. Entorno de diseño: componentes etiquetados

A continuación, se va explicar el proceso de diseño de la interfaz gráfica externa y cómo se ha ido desarrollando y utilizando todas las herramientas de la paleta y propiedades de configuración para cada elemento.

1. Se ha coloreado el fondo de la interfaz gráfica utilizando el menú "Property Inspector" de la barra de herramientas y seleccionando el color deseado en el menú contextual de las propiedades de la figura. Además, se ha activado la opción "ToolBar" para disponer de la paleta de herramientas gráficas a la hora de ejecutar la interfaz gráfica externa. Véase *Figura 35*.

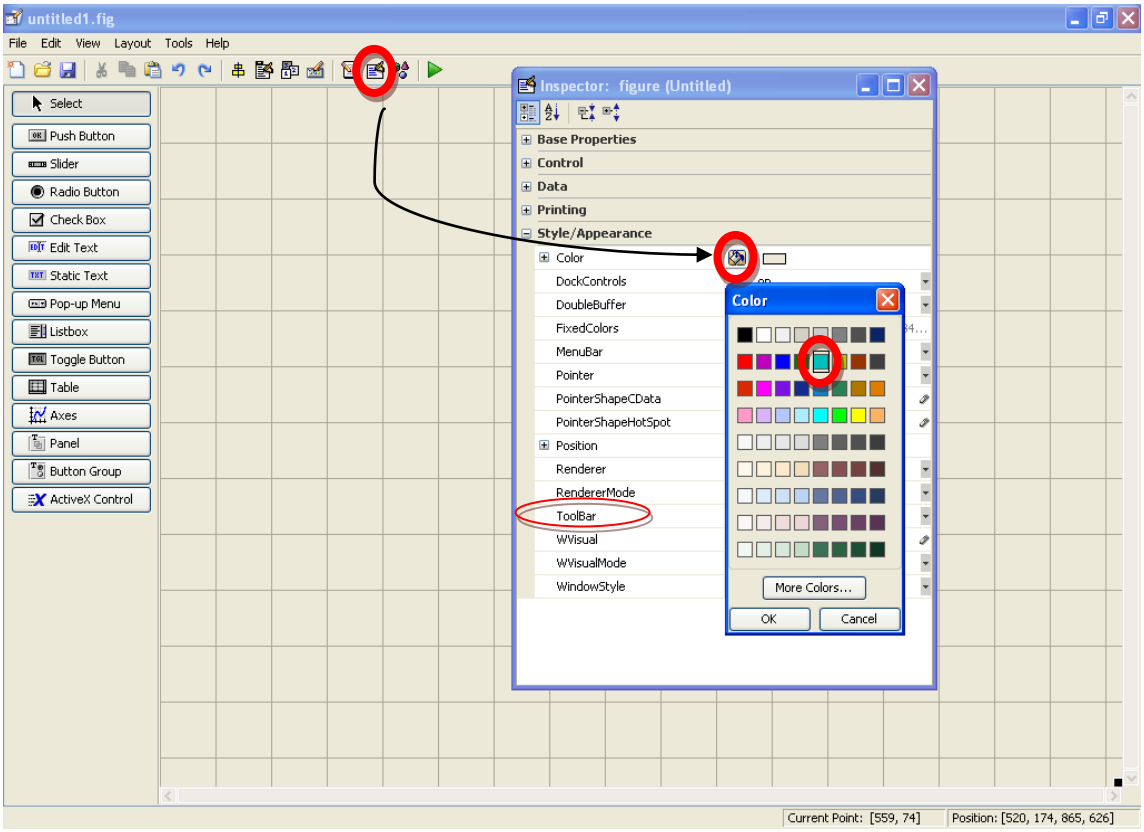


Figura 35. Configuración del color del fondo de pantalla

2. Se han añadido todos los paneles que forman el diseño de la interfaz gráfica y se han configurado mediante la opción "Property Inspector" de la barra de herramientas para pintarlos del color deseado, incrementar su relieve y escribir el título o encabezado de cada panel en caso de que sea necesario.

   Un panel tampoco no es un control propiamente dicho. Su función es la de englobar una serie de opciones (botones, cajas de texto, etc.…) con el fin de mantener una estructura ordenada de controles, separando unos de otros en función de las características del programa y del gusto del programador.

A continuación, se visualiza el orden y distribución de todos los paneles creados en la interfaz gráfica y las opciones seleccionadas para su configuración. El nombre del panel seleccionado se configura mediante la opción "Title" del menú contextual de propiedades y para el relieve se selecciona la opción "Border Type" y se elige el tipo "beveledout". Véase *Figura 36*.

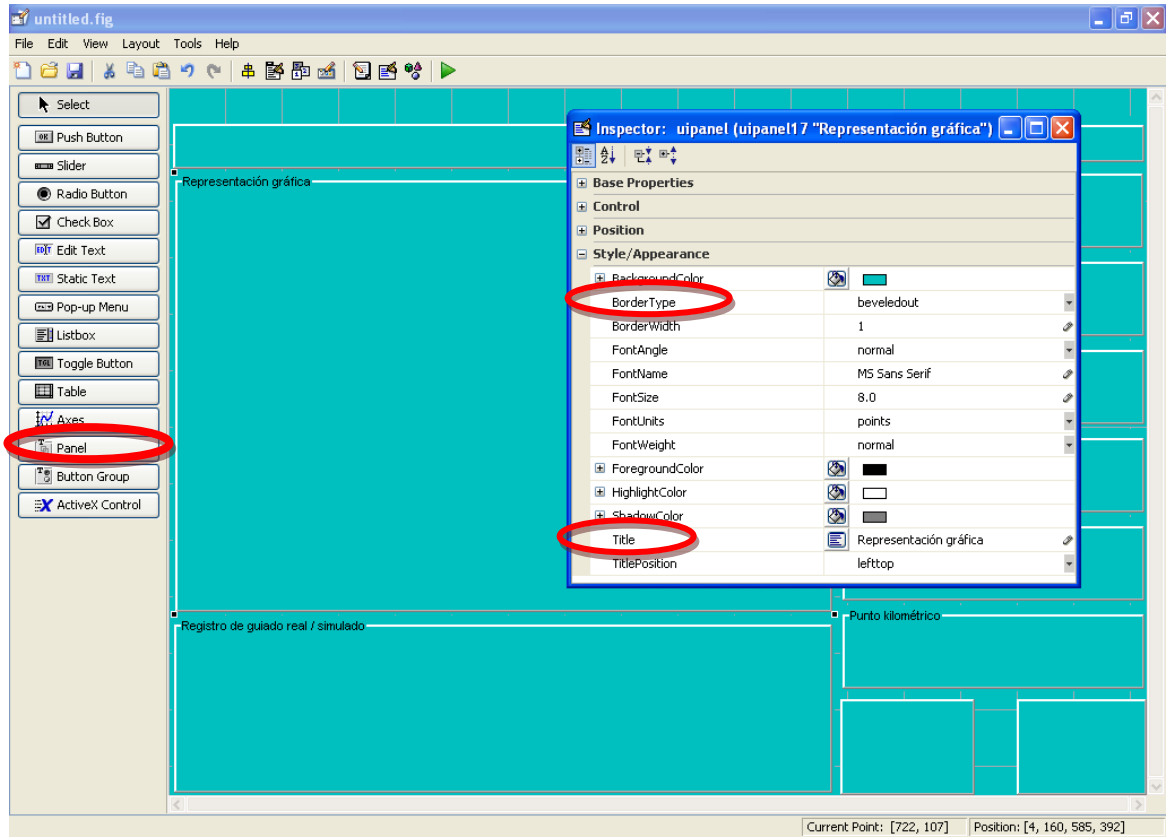


Figura 36. Configuración del título y de los paneles

Para poder realizar una alineación y distribución homogénea de todos los paneles que se han creado en la interfaz, se seleccionan los que se desean ordenar y se pulsa el icono "Align Objects" en la paleta de herramientas, desplegándose un menú contextual con variedad de opciones a elegir. Véase *Figura 37*.

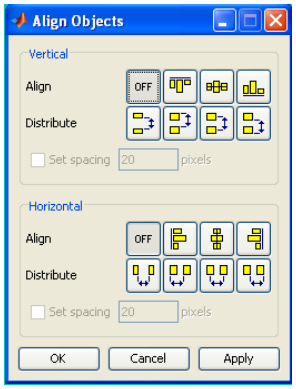
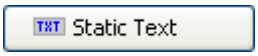


Figura 37. Alineación de los componente de la interfaz gráfica

3. Se incluyen todos los textos estáticos de los paneles de control y del título de la interfaz gráfica. Cada nombre de variable, cada indicador numérico y cada unidad de magnitud es un texto estático independiente correctamente alineado mediante la opción anteriormente explicada "Align Objects". Véase *Figura 38*.



Los textos estáticos no son controles, ya que no permite realizar ninguna operación con el ratón. Permiten escribir un texto en una caja en la pantalla.

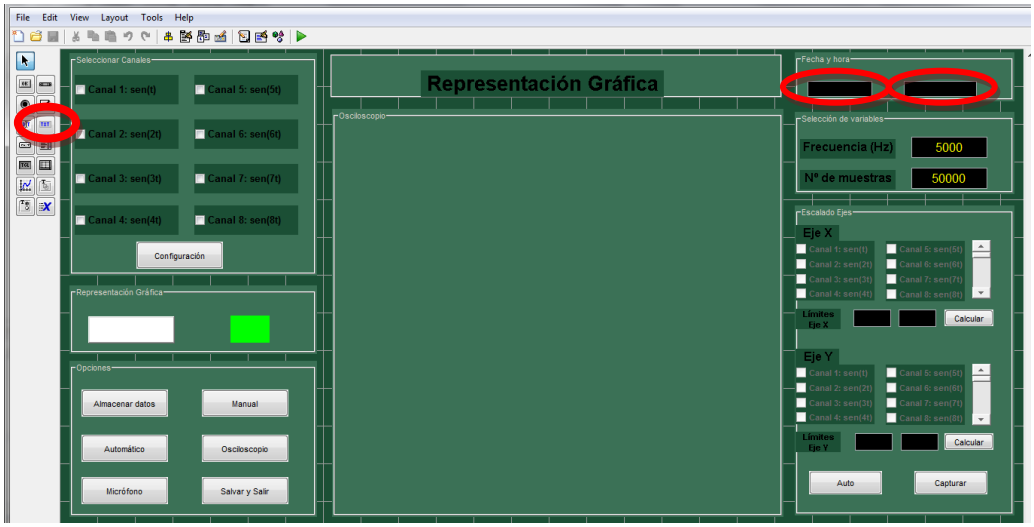


Figura 38. Colocación de los textos estáticos

Para realizar el diseño gráfico de los indicadores numéricos, se configuran sus opciones de formato seleccionando el indicador en cuestión y pulsando el icono de la paleta de herramientas "Property Inspector". Véase *Figura 39*.

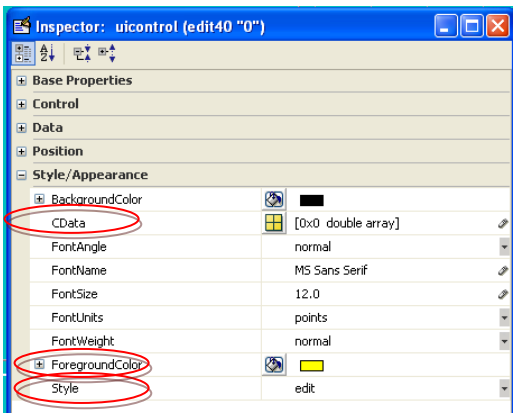


Figura 39. Configuración del texto estático

Se configura el color de fondo negro del texto estático mediante la opción "BackgroundColor" y el color amarillo de los números a través de la opción "ForegroundColor". Por último, se cambia el estilo mediante la opción "Style" y se elige el tipo "edit" para ahondar el indicador numérico y dar más sensación de realismo.

4. Se introducen mediante el componente de la paleta "Push Button" los cuatro botones de interacción de la interfaz gráfica.

Los botones son pequeños objetos de la pantalla normalmente acompañados con texto. Al pulsar sobre ellos con el ratón genera una acción y su rutina de llamada se ejecuta. Véase *Figura 40*.

Figura 40. Botones de la interfaz gráfica

Se configura el formato del texto del interior del botón del mismo modo que para los componentes anteriores y se cambia el nombre del botón mediante la opción "String". Véase *Figura 41*.

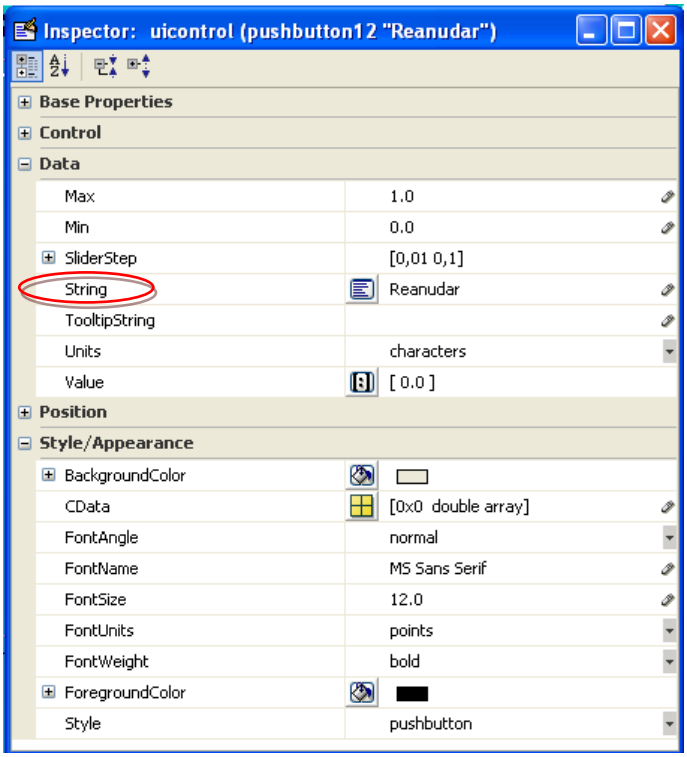
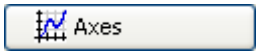


Figura 41. Configuración de los botones de la interfaz gráfica

5. Se introduce en la interfaz los ejes para la representación gráfica pulsando en la paleta de componentes la opción “Axes”.



Crea un área donde se pueden hacer representaciones gráficas de coordenadas tanto en dos dimensiones como en tres dimensiones, se puede añadir una cuadrícula, configurar la escala numérica y escribir etiquetas de los ejes.

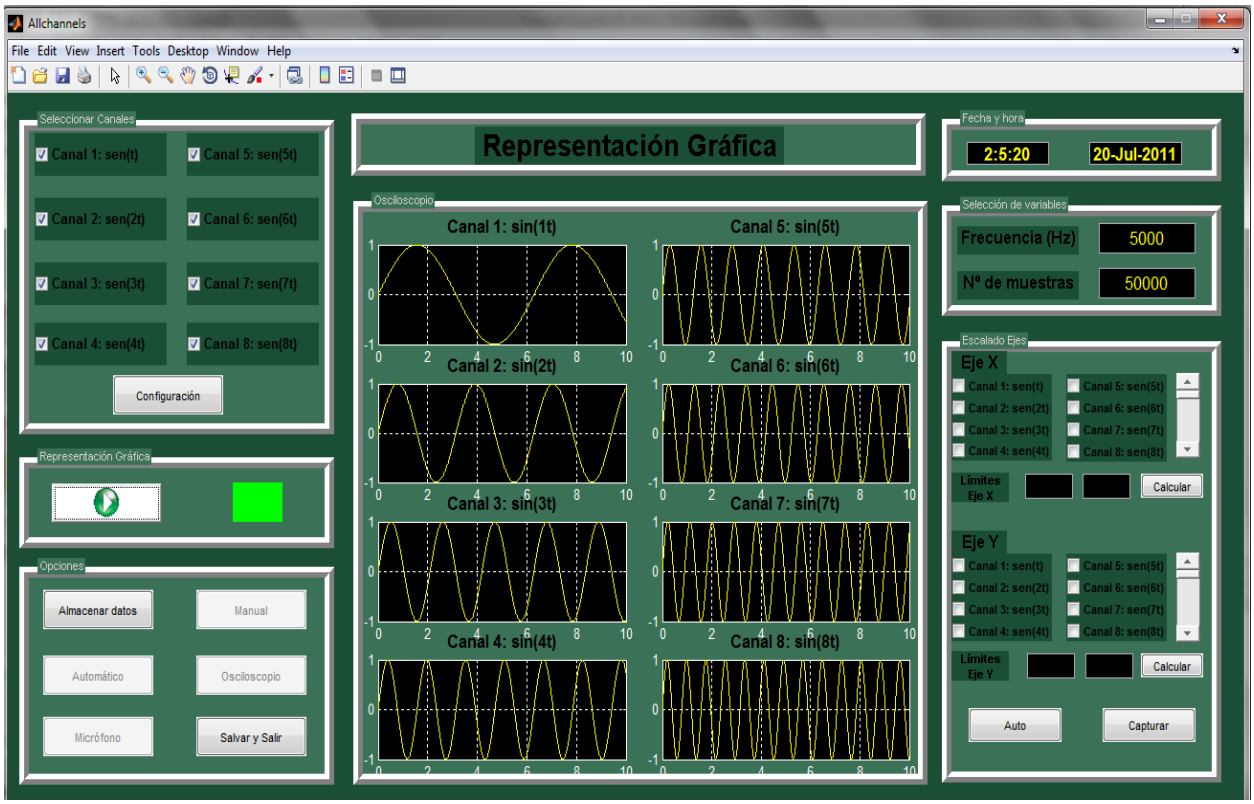


Figura 42. Interfaz gráfica de usuario

Para realizar la configuración de las propiedades de los ejes, se selecciona el componente ejes y se pulsa sobre el icono “Property Inspector” para desplegar un menú contextual con todas las opciones tanto gráficas como paramétricas de los tres ejes (X, Y, Z).

A continuación, se muestra el menú de propiedades disponible para el componente "Axis". Véase *Figura 43*.

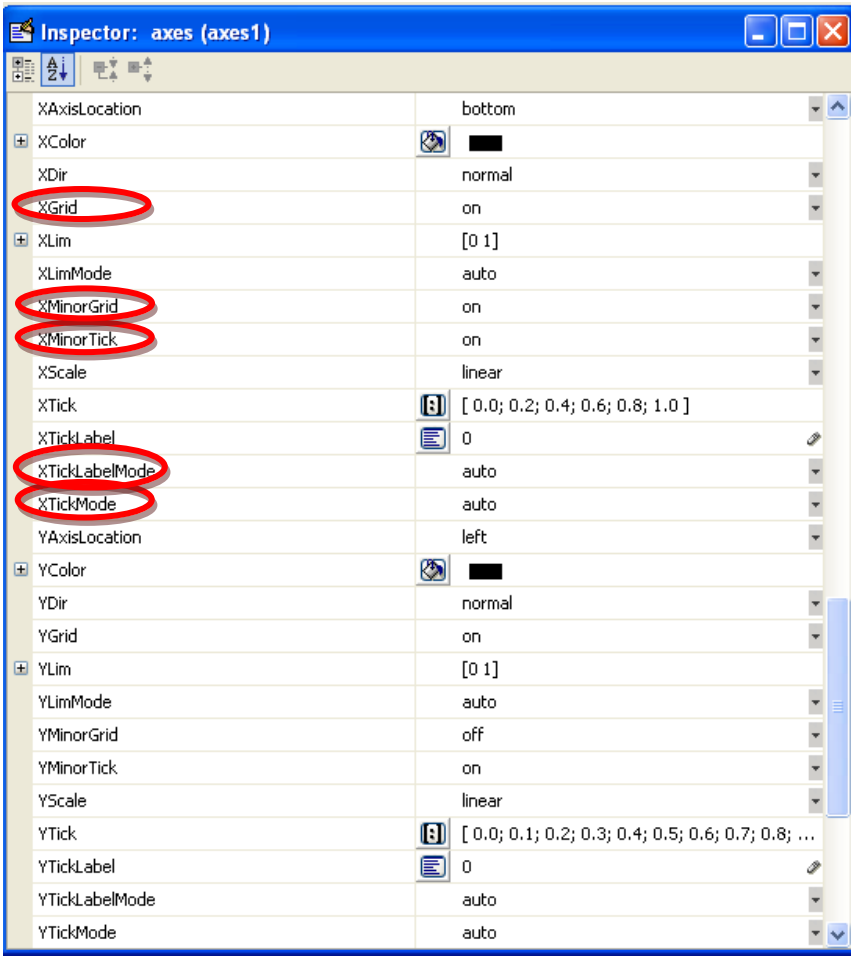


Figura 43. Configuración del eje de coordenadas de la interfaz gráfica

Algunas de las propiedades más importantes que se deben configurar para optimizar la representación gráfica de las coordenadas son las que respectan a la escala de los ejes y a su punto de vista.

- "XGrid" → Dibuja una cuadrícula en el eje X.
- "XMinorGrid" → Dibuja una cuadrícula mucho más habitada en el eje X.
- "XLimMode" → Permite alternar entre modo automático y manual para introducir los límites superior e inferior en el eje X.
- "XTickLabelMode" → Permite alternar entro modo automático y manual para introducir las etiquetas de numeración del eje X.
- "XTickMode" → Permite alternar entre modo automático y manual para introducir la escala de numeración del eje X.

6. Se insertan los menús interactivos en la interfaz gráfica mediante el icono de la barra de herramientas "Menu Editor". Esta herramienta permite crear varios menús de forma jerarquizada. Véase *Figura 44*.

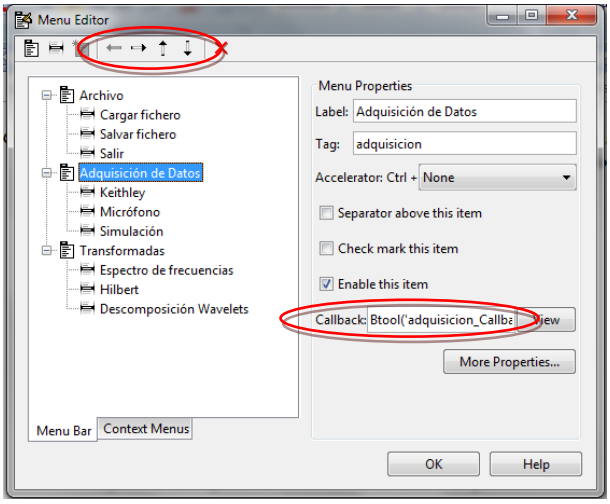


Figura 44. Configuración de los menús de la interfaz gráfica

Mediante las flechitas colocadas en la parte superior de la figura anterior, se puede mover el menú seleccionado a un nivel superior o inferior. Además se le puede añadir una tecla de acceso rápido configurando la opción "Accelerator".

7. Por último, se guarda los cambios realizados en la interfaz gráfica y se ejecuta pulsando el icono "Run" de la barra de herramientas.

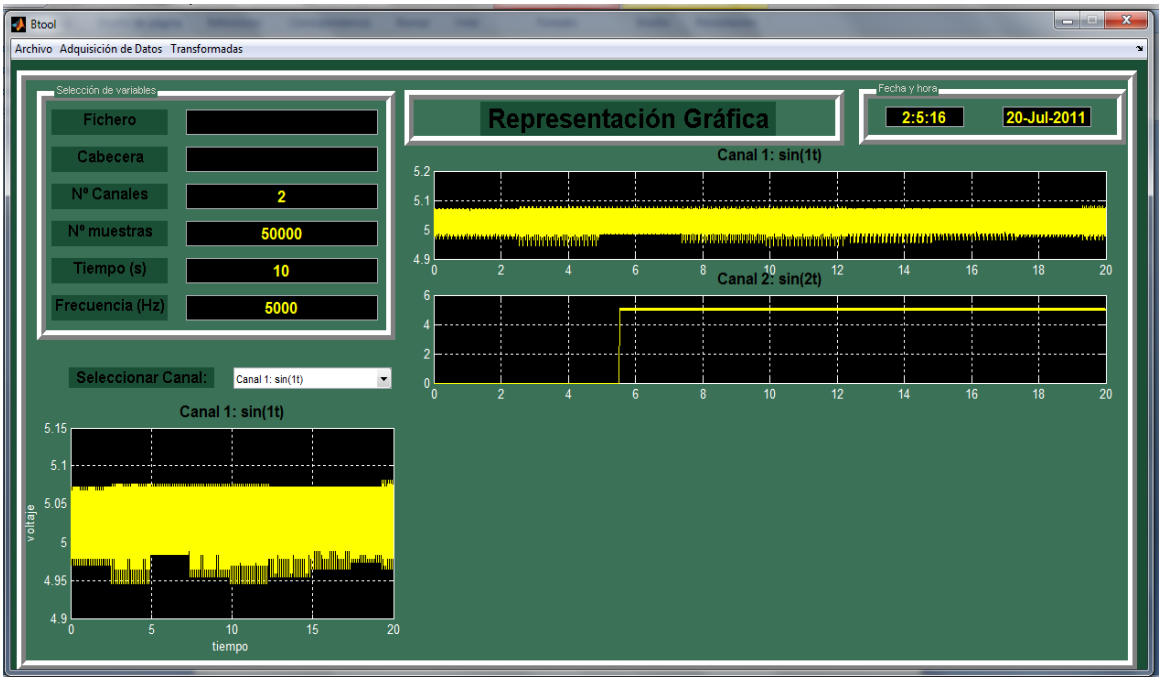


Figura 45. Presentación de la GUI en ejecución

En la figura anterior, se muestra la interfaz de guiado después de diseñar y configurar todos los componentes y de lanzar su ejecución. Véase *Figura 45*.

3.3.5 Funcionamiento de una aplicación GUI

Una aplicación GUIDE consta de dos archivos: **.m* y **.fig*.

- El archivo **.m* es el que contiene el código con las correspondencias de los elementos de control de la interfaz gráfica. Cada vez que se introduzca un elemento gráfico en el panel de diseño de la interfaz (**.fig)* se generará unas líneas de programa automáticamente asociadas a ese tipo de control. Estas líneas de programas están vacías, es decir, es necesarios programar acciones a elementos de control para llevarlas a cabo durante la ejecución del programa.

- El archivo **.fig* es el que contiene los elementos gráficos así como las propiedades de la interfaz. Cada vez que se adicione un nuevo elemento en la interfaz gráfica, se genera automáticamente código en el archivo **.m*.

Para ejecutar una interfaz gráfica, si la hemos etiquetado con el nombre de *ejemplo.fig*, simplemente se ejecuta en la ventana de comandos

>> *ejemplo*

O bien, haciendo clic derecho en el **.m* correspondiente y seleccionando la opción "Run". Véase *Figura 46.*

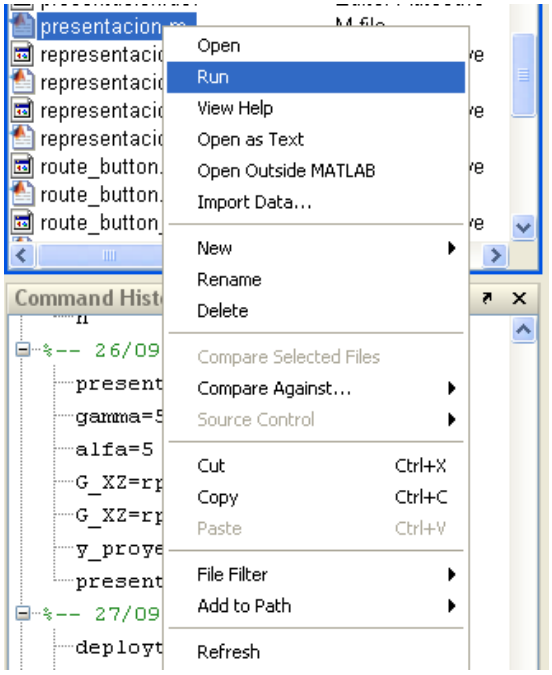


Figura 46. Ejecutar una GUI desde el directorio de trabajo

Además, para entender el funcionamiento de una aplicación GUI es necesarios desarrollar la jerarquía que existe entre los diferentes elementos de MATLAB. Los gráficos en MATLAB tienen una jerarquía formada por objetos de distintos tipos.

Esta jerarquía tiene una distribución de árbol con el aspecto como el que se muestra en la *Figura 47*.

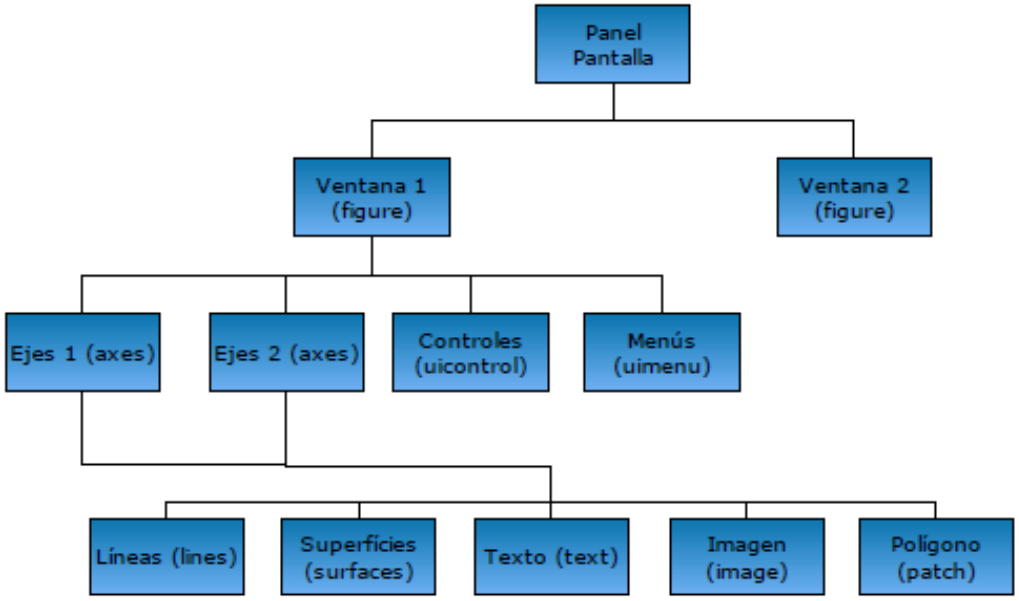


Figura 47. Diagrama de bloques de la jerarquía de MATLAB

El objeto más general es la pantalla o panel, es la raíz de todos los demás y solo puede haber un objeto pantalla. Una pantalla puede contener una o más ventanas (figures).

A su vez cada una de las ventanas puede tener uno o más ejes de coordenadas (axes) en los que se puede representar objetos de más bajo nivel. Una ventana puede también puede tener también controles (uicontrols) como botones de selección o de opción y también menús.

Finalmente, los ejes pueden contener los cinco tipos de elementos gráficos que permite MATLAB: líneas (lines), polígonos (patch), superficies (surfaces), imágenes tipo bitmap (image) y texto (text). La jerarquía de objetos indica que en MATLAB hay objetos padre e hijos.

Por ejemplo, el objeto ventana es hijo de pantalla, y a su vez cada objeto ventana es padre de los ejes. En el momento que se borre un objeto de MATLAB se borrarán todos los objetos que son descendientes. Es decir, al borrar el objeto eje se borrarán las líneas, superficies, texto, imagen y polígono.

Los objetos de MATLAB pueden tener distintas propiedades y algunas de las propiedades de los objetos más importantes son:

- Children
- Clipling
- Parent
- Type
- UserData
- Visible

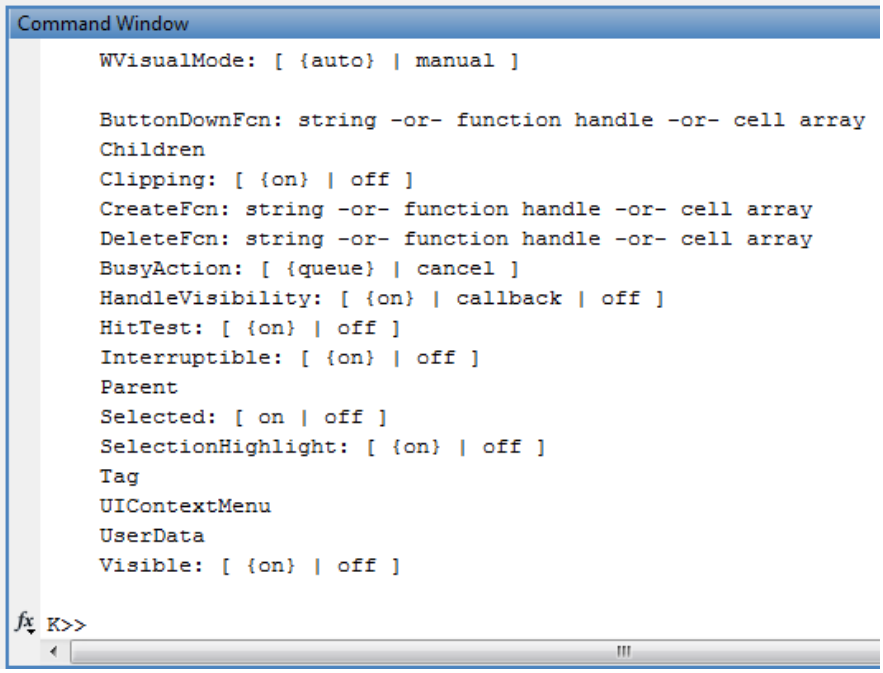


Figura 48. Propiedades de los objetos del un identificador

Las propiedades tienen valores por omisión que se utilizan cuando el usuario no indique otra cosa. Es posible cambiar las propiedades por omisión y también devolverles el valor original.

Hay propiedades que solo pueden ser consultados sus valores, sin poder ser modificados y otros que tendrán un conjunto limitado de valores, por ejemplo, on/off.

3.3.6 Flujo de operación

El concepto básico de la operación del software con una interfaz gráfica de usuario es que el flujo de cómputo está controlado por las diferentes acciones en la interfaz.

Mientras que en un guión el flujo de comandos está predeterminado, el flujo de operaciones con una GUI no lo está. Los comandos para crear una interfaz con el usuario se escribe en un guión, la interfaz invoca el guión que se ejecute, mientras la interfaz del usuario permanece en la pantalla aunque no se haya completado la ejecución del guión.

Cuando se interactúa con un control, el programa registra el valor de esa opción y ejecuta los comandos prescritos en la cadena de invocación.

El control guarda un *string* que describe la acción a realizar y cuando se invoca puede consistir en un solo comando de MATLAB, en una secuencia de comandos o en una llamada a una función. Es recomendable utilizar llamadas a funciones, sobre todo cuando se requieren de más de unos cuantos comandos en la invocación.

Los menús de interfaz con el usuario, los botones, los menús desplegables, los controladores deslizantes y el texto editable son dispositivos que controlan las operaciones del software. Al completarse la ejecución de las instrucciones de la cadena de invocación, el control vuelve a la interfaz para que puedan elegirse otra opción del menú. Este ciclo se repite hasta que se cierra la interfaz gráfica.

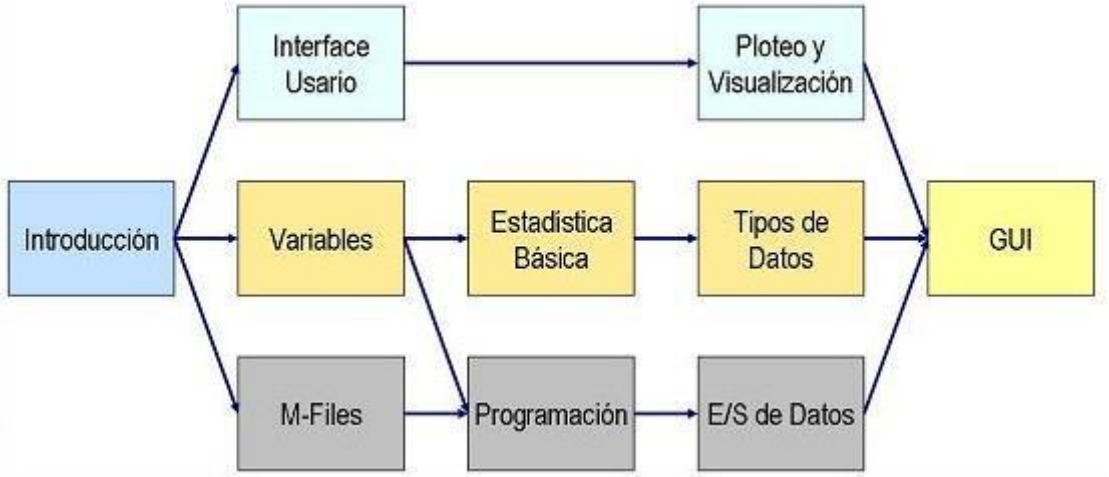


Figura 49. Ciclo de operación de la GUI

En la figura anterior, se muestra un esquema gráfico del flujo de operación que hay que tener en cuenta a hora de diseñar y desarrollar una interfaz gráfica externa. Véase *Figura 49*.

3.3.7 Alternativas de comunicación entre aplicaciones MATLAB

3.3.7.1 Variables globales

Para realizar una comunicación entre dos programas diferentes de MATLAB mediante variables globales, en primera instancia puede parecer una opción sencilla y accesible [16].

Las variables definidas dentro de una función son variables locales, es decir, las variables que se crean dentro de la función pertenecen al espacio de trabajo de la función y son inaccesibles desde otras partes del programa. Además no interfieren con variables del mismo nombre definidas en otras funciones o partes del programa y desaparecen una vez finaliza la llamada a la función.

Por tanto, para que la función se pueda comunicar y tener acceso a variables que no han sido pasadas como argumentos es necesario declarar dichas variables como variables globales, tanto en el programa principal como en las distintas funciones que deben acceder a su valor. Efectivamente, su valor es visible en todos los espacios de trabajo de las funciones donde han sido declaradas. Es decir, el ámbito de una variable global son todas las funciones que componen el programa, cualquier función puede acceder a dichas variables para leer y escribir en ellas y además se puede hacer referencia a su dirección de memoria en cualquier parte del programa.

Sin embargo, este método no es válido para realizar una comunicación de variables entre diferentes programas. La razón por la cual no se puede llevar a cabo la comunicación es porque al crear los ejecutables externos (**.exe*) de cada uno de los programas que se quieren comunicar, el espacio de trabajo donde se almacenan todas las variables globales deja de ser común a ambos programas al trabajar de forma externa a MATLAB.

Por ese motivo, trabajar desde el ámbito de MATLAB para realizar intercambios de variables resulta sencillo y cómodo porque las variables globales están al alcance en cualquier momento, pero para conseguir realizar una comunicación mediante aplicaciones externas es necesario recurrir a otros métodos más certeros.

3.3.7.2 Fichero de intercambio de datos (.mat)*

Se basa en la creación y uso de un fichero compartido que permita leer y escribir tanto el nombre de la variable como su valor, siendo accesible en cualquier momento por cualquiera de los ficheros GUI que están comunicados entre ellos.

No presenta el problema de tener varios programas en diferentes carpetas de trabajo, puesto que se puede hacer la llamada al archivo indicando previamente la ruta o "path" donde está ubicado, beneficiando la independencia de cada programa.

Este tipo de archivo con la extensión **.mat*, no sólo tiene comandos específicos para su escritura y lectura, sino que además mantiene el formato que tengan las variables. Esto beneficia enormemente la programación del software ya que no es necesario tener que especificar el tipo de formato junto al valor a la hora de escribirlo o leerlo. Estas ventajas no se obtienen en otros tipos de archivos comúnmente utilizados, como los archivos de texto plano **.txt* o de tabla **.xls.*

A continuación, se va describir brevemente los comandos utilizados para realizar la comunicación y su funcionalidad.

El comando "save" permite guardar el valor de todas las variables que se deseen almacenar en el fichero **.mat*, tan sólo especificando los nombres de las variables consecutivamente. Además, mediante la instrucción "-append" al final del comando "save", crea variables nuevas en el archivo en caso de que no hayan sido almacenadas anteriormente o sobrescribe su valor si ya habían sido almacenadas. El uso de la instrucción "-append" evita que se borren otras variables guardadas en el archivo de intercambio ya que no todos los ficheros harán uso de las mismas variables del archivo.

```
save datos.mat x y alfa beta gamma acceso -append
```

El comando "load" lee el valor de las variables y las guarda en el archivo de intercambio con el mismo nombre si están siendo usadas por el programa. Sin embargo, las genera nuevas si no han sido declaradas todavía. Al igual que el comando "save", nos permite hacer una lectura selectiva de las variables especificando el nombre de las variables elegidas consecutivamente, sin la necesidad de tener que cargar todas las variables, ocupando espacio en memoria innecesario y evitando posibles pérdidas de valores.

```
load datos.mat x y z
```

Por tanto, para la comunicación de variables, vectores o matrices entre dos ficheros **.m* ya sean funciones o scripts, se realiza mediante la creación de un fichero común con extensión **.mat*.

Una dificultad añadida a la hora de compartir un mismo archivo para guardar las variables de intercambio, es la necesidad de regular el tráfico de los programas al acceder por medio de lectura o escritura al archivo.

3.3.7.3 Socket

La comunicación por Socket es comúnmente usada para realizar el intercambio de datos en una red Ethernet bajo el tipo cliente-servidor. Más exactamente, un socket es un punto de comunicación por el cual un proceso puede emitir o recibir información.

Este tipo de comunicación necesita ser configurado mediante una dirección IP, un protocolo de transporte y un número de puerto, por lo que está muy orientado a la comunicación para programas que se ejecuten en diferentes máquinas.

En el caso de nuestro sistema no es necesario, puesto que se ejecutarán todos los programas en el mismo ordenador, por lo que no es a priori una ventaja frente a otros métodos puesto que además obliga a realizar la configuración de los puertos de red.

El inconveniente que supone tener que dar formato a los valores obtenidos en esta comunicación, en función del dato que se vaya a recibir, ha sido fundamental a la hora de desestimar este método de comunicación frente al elegido.

3.4 Análisis y adquisición de datos

Un sistema de Adquisición de Datos (AD) se usa cuando se está interesado en medir y analizar algún fenómeno físico. Por tanto, es una colección de herramientas HW/SW que nos permite interactuar con el mundo físico permitiendo mediante un equipo tomar señales físicas del entorno y convertirlas en datos que posteriormente podremos procesar y presentar.

Se puede adquirir, visualizar y salvaguardar señales utilizando la función "Scopesoft", trabajando en modo externo. Quiere decir que se puede observar el comportamiento del modelo en tiempo real y guardar los datos en el Workspace.

Existen dos modalidades de captura y visualización de las señales:

- **Signal Tracing:** proceso mediante el cual se puede adquirir y visualizar señales durante la ejecución de una aplicación en tiempo real. Es decir, permite visualizar los datos mientras que los está capturando, sin tener que esperar a que acabe la simulación.
- **Signal Logging:** proceso mediante el cual se puede adquirir y visualizar señales procedentes de la aplicación en tiempo real, una vez haya acabado la ejecución o bien se haya parado manualmente. Hasta que no se cumpla alguna de las dos condiciones no se permite visualizar, guardar y/o exportar las señales.

La tarjeta de adquisición de datos permite capturar y/o generar señales reales e interactuar con ellas desde la aplicación en tiempo real. Un sistema de adquisición de datos está formado por un hardware y software que permite a un sistema digital conectarse al mundo real. El sistema de adquisición de datos está formado por:

- Hardware de adquisición de datos.
- Sensores y actuadores.
- Hardware de acondicionamiento de señal.
- Computadora o procesador.
- Software.

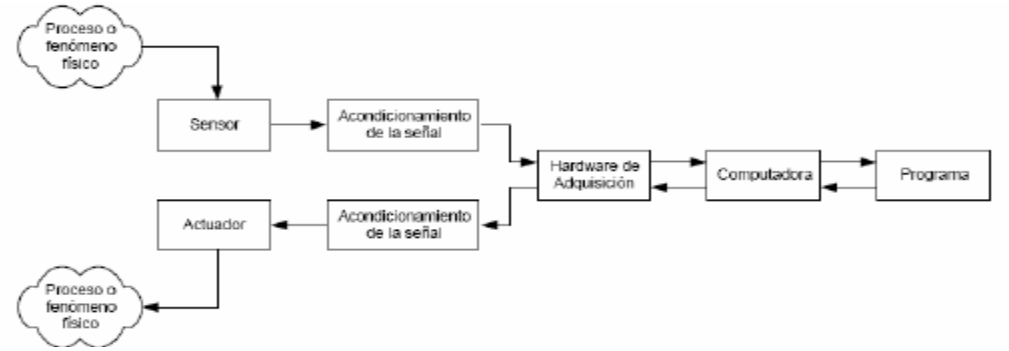


Figura 50. Esquema del proceso de adquisición de datos

3.4.1 Tarjeta de adquisición de datos

Hardware de adquisición de datos

Es el corazón de cualquier sistema de adquisición de datos. Su función es convertir señales analógicas provenientes del mundo real a señales digitales, o bien convertir señales digitales a analógicas [11]. Véase *Figura 51*.

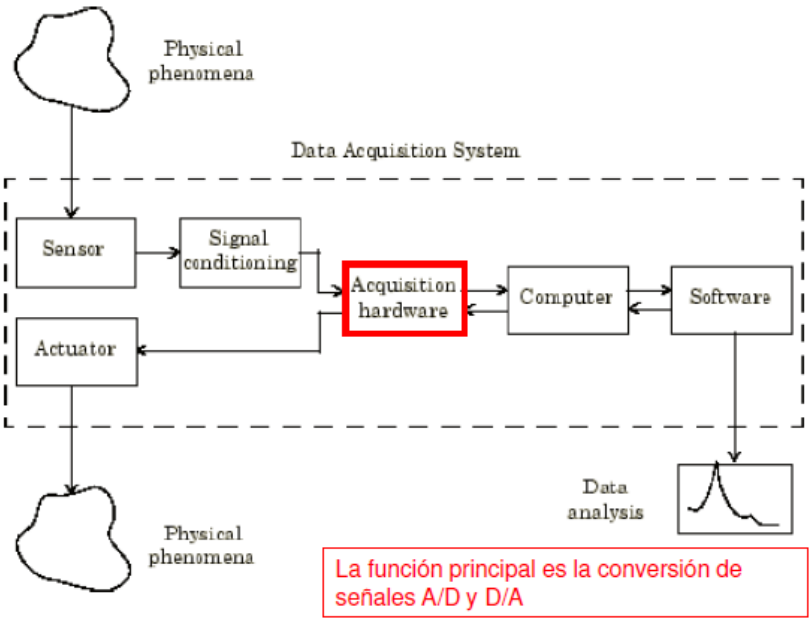


Figura 51. Esquema del Hardware de adquisición de datos

El hardware de AD puede presentarse de dos maneras:

- Interna e instalada directamente en un ranura de expansión dentro de la computadora.
- Externa que se conecta a la computadora a través de una cable externo.

Se caracteriza por lo subsistemas que posee donde un subsistema es un componente que realiza una tarea específica. Existen 4 subsistemas:

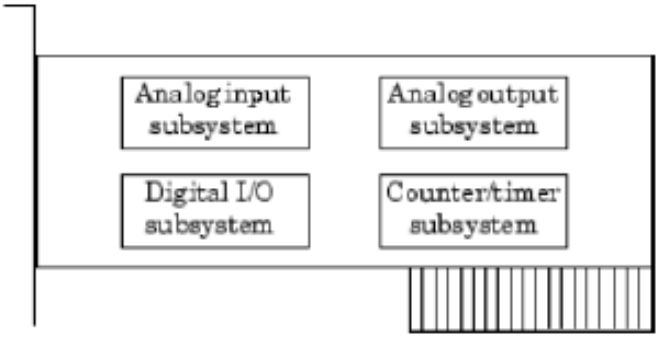


Figura 52. Subsistemas del hardware AD

Sensores y actuadores

Los sensores y actuadores son aquellos que actúan como transductores, es decir estos elementos transforman una señal capturada de una naturaleza en otra señal de salida de otra naturaleza. Por ejemplo, un sensor sería el tacómetro que lee las vueltas que realiza el eje del motor y genera una señal eléctrica proporcional. El actuador es el que mediante una señal eléctrica hace que el motor gire a determinada velocidad. Véase *Figura 53*.

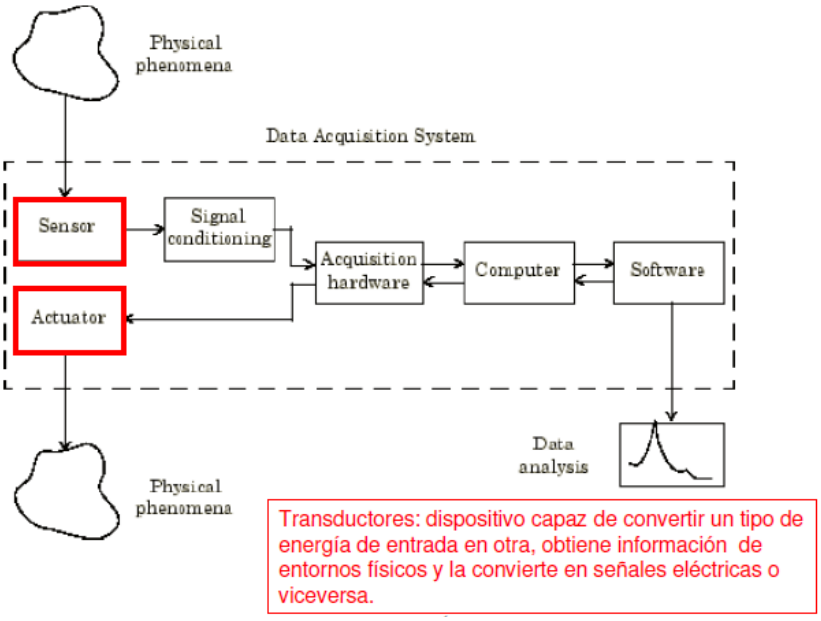


Figura 53. Sensores y actuadores de un sistema de adquisición de datos

Los sensores se pueden clasificar por:

- **Sensores Digitales**: Switches, Encoders, Botones, etc.
- **Sensores Analógicos:** Acelerómetros, Micrófonos, medidores de presión, medidores de temperatura, etc.

Las dos características más importantes de los sensores son:

- **Salida del sensor:**
  - Salidas digitales o analógicas.
  - Salidas de corriente: 4 – 20 mA, uso de una resistencias de precisión para convertir a voltaje.
  - Salidas de voltaje: caracterizadas por Amplitud, Frecuencia y Duración.
- **Ancho de Banda del sensor:**
  - Se refiere a las frecuencias presentes en la señal que está siendo medida.
  - Se puede pensar también como la razón de cambio de la señal.

Hardware de acondicionamiento de señal

Normalmente las señales de los sensores son incompatibles con el hardware de adquisición de datos. Para lograr esta compatibilidad habrá que acondicionar la señal, es decir que si la señal es muy pequeña habrá que amplificarla, de lo contrario atenuarla. Véase *Figura 54*.

También es muy común eliminar componentes frecuenciales de las señales.

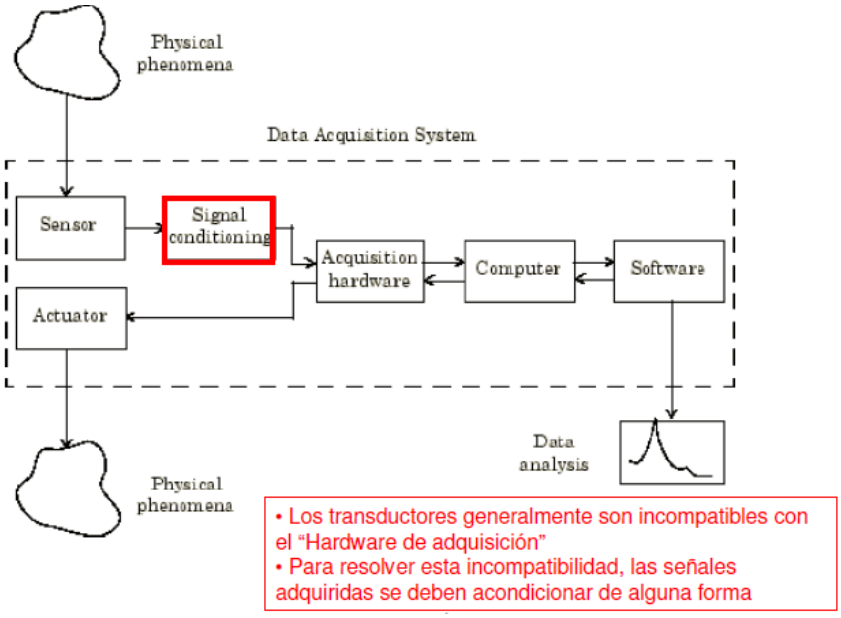


Figura 54. Acondicionamiento de señal en un sistema de adquisición de datos

El acondicionamiento se usa por la "incompatibilidad" de las señales adquiridas.

Los tipos de acondicionamiento son:

- **Amplificación:** niveles bajos de entrada (100 mvolts) necesitan ser amplificados.
- **Filtrado:** eliminación de ruido de la señal de interés.
- **Ajuste:** ajustar el rango del transductor al del convertidor A/D.
- **Conversión:** transformar la información para que sea siempre voltaje y que las corrientes sean proporcionales a voltajes.
- **Multiplexación:** técnica que permite mandar distintas señales sobre un mismo canal.
- **Acondicionamiento de impedancias:** cuando se conectar un segundo circuito a la salida de un primero, hace que la tensión de salida del primero se vea modificada.

Computadora o procesador

La computadora proporciona un procesador, un reloj, un bus para transferir datos y espacio de memoria o disco para almacenar datos. Véase *Figura 55*.

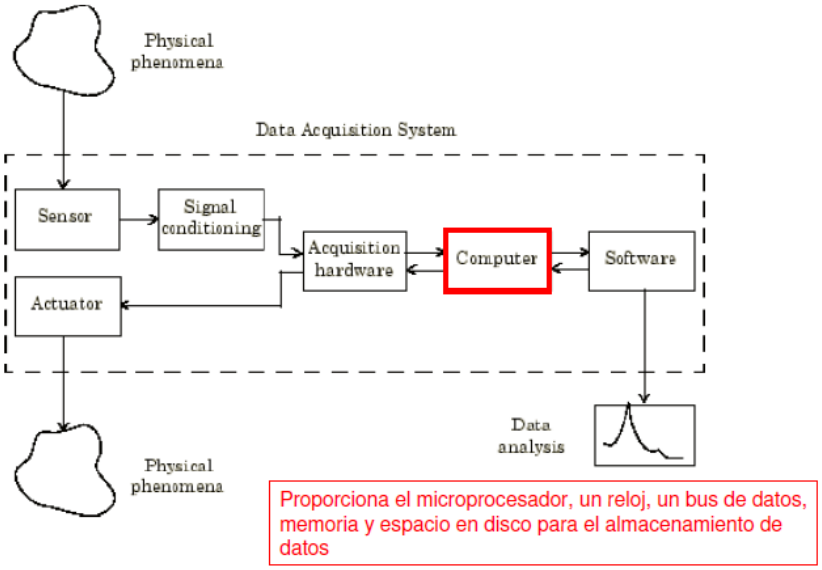


Figura 55. Procesador de un sistema de adquisición de datos

Software

El software de adquisición de datos permite intercambiar información entre la computadora y el hardware. Por ejemplo, los programas típicos permiten configurar la tasa de muestreo de una tarjeta de adquisición y adquirir un número concreto de muestras. Véase *Figura 56*.

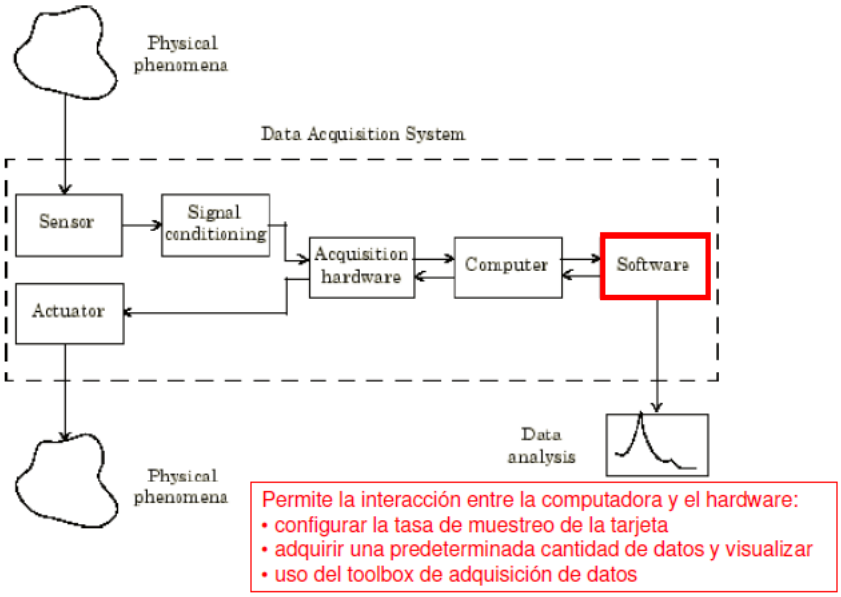


Figura 56. Software de un sistema de adquisición de datos

Hay dos clases de Software:

- **Driver SW:** accede y controla las capacidades del hardware.
- **Application SW:** funciones de alto nivel para realizar aplicaciones.

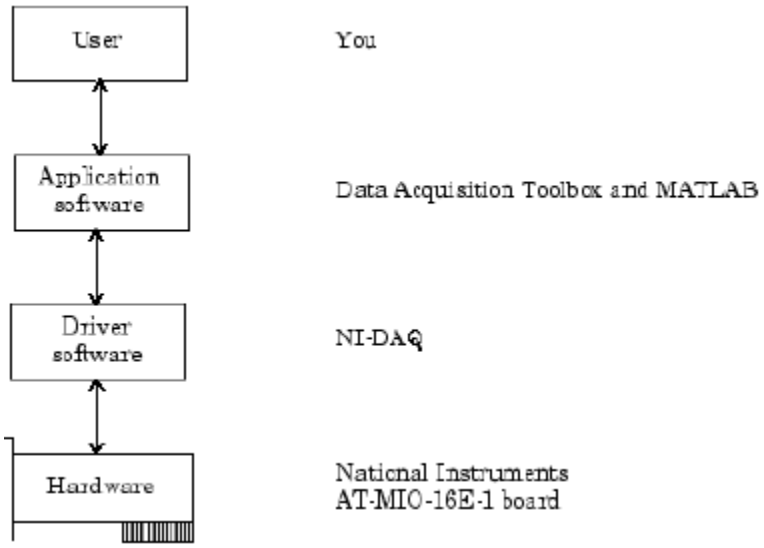


Figura 57. Clases de software

Dos cosas importantes a resaltar:

- Las entradas son adquiridas por un sensor, se acondicionan, después se convierten en 'bits' para que la computadora las pueda leer y son manipuladas o analizadas para extraer información relevante.
- Los datos desde la computadora son transformados en señales analógicas y son pasados hacia el 'exterior' por medio de un actuador.

Pasos a realizar para la adquisición de datos

1. Crear un objeto tipo dispositivo.
2. Agregar canales (E/S analógicas) o líneas (E/S digitales).
3. Configurar propiedades.
4. Colocar en fila datos (solo para Salidas Analógicas).
5. Arrancar la adquisición de datos (o sacar datos).
6. Esperar a que la adquisición se complete.
7. Extraer los datos adquiridos (solo para entradas analógicas).
8. Finalizar la sesión (limpiar variables del workspace y de memoria).

3.4.2 Comunicación USB con el PC

Para poder llevar a cabo la comunicación entre los diferentes módulos, es necesario el desarrollo tanto del hardware de todos los componentes pertenecientes al sistema como del software de los tres módulos relacionados con el ordenador central, tarjeta KEITHLEY, tarjeta sonido y simulador. Véase *Figura 58*.

La conexión USB (Universal Serial Bus) es una interfaz de conexión de dispositivos periféricos a un PC o host similares. Una de sus posibles aplicaciones es la medición y control de sistemas o como la necesitada por un dispositivo de propósito general que permita sensar o realizar una acción. USB es un bus punto a punto, con inicio en el host y destino en un dispositivo o Hub, los cuales según este protocolo de comunicación pueden llegar a ser máximo 127 dispositivos.

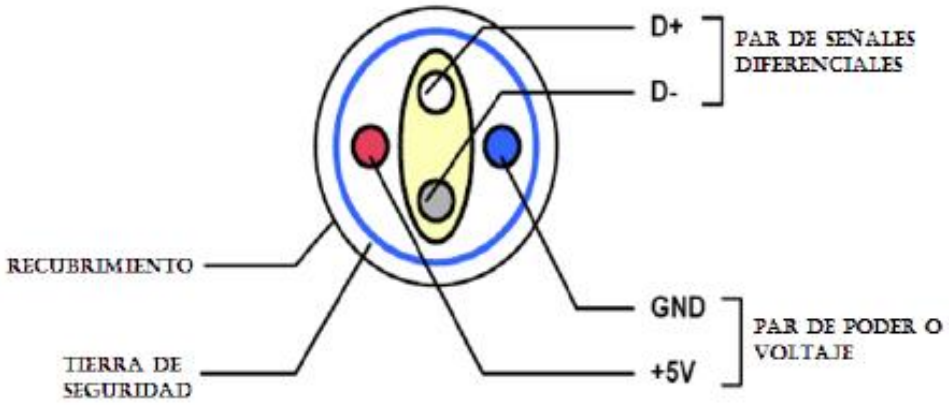


Figura 58. Esquema de conexión del puerto USB

Físicamente la transmisión se realiza por un par trenzado (D y –D), y requiere dos pines más para suministrar el voltaje necesario a la conexión o pequeños dispositivos periféricos. En este protocolo solo puede existir un único host, que es el dispositivo maestro, que inicializa la comunicación y el Hub, que es el dispositivo que contiene uno o más conectores o conexiones a otros dispositivos USB.

El protocolo de comunicación se basa en el paso de testigo (token), donde el host proporciona el testigo al dispositivo seleccionado y este le devuelve el testigo como respuesta.

Es bien sabido, que los microcontroladores han dado la pauta en la electrónica en cuanto a realizar el monitoreo y control de dispositivos, dado lo robusto del dispositivo y la cantidad de memoria que posee. Sin embargo, el poder conectar un microcontrolador con un PC, es decir, con un microprocesador, trabajando en conjunto supera en gran medida las expectativas, además de permitir al usuario tener un ambiente amigable para intervenir en el sistema.

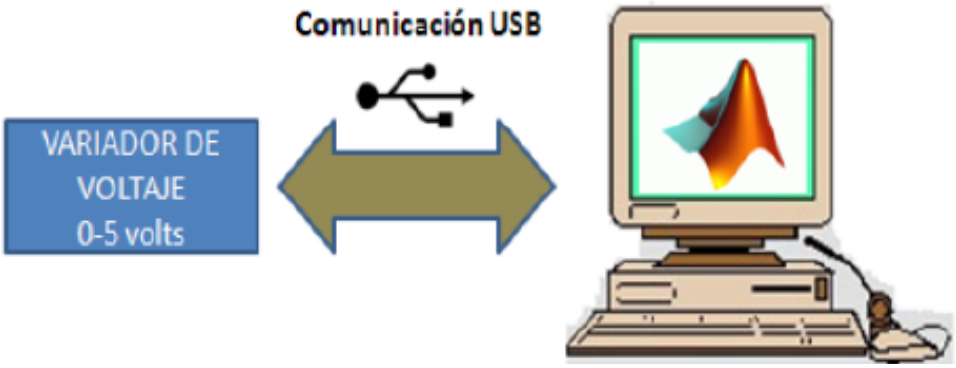


Figura 59. Interface gráfica en MATLAB usando comunicación USB

3.4.3 Tarjeta de sonido integrada en el PC

De la misma manera que con el puerto paralelo, el control de la tarjeta de sonido se realiza mediante la toolbox de adquisición de datos de MATLAB el cual se divide en tres componentes principales:

- Las funciones M establecidas.
- El motor de adquisición de datos.
- Los manejadores (drivers) de la tarjeta de adquisición de datos (en este caso, la de sonido).

Como se muestra en la *Figura 60*, estos componentes permiten intercambiar información entre MATLAB y el hardware de adquisición de datos.

Figura 60. Componentes del toolbox de adquisición de datos

Propiedades

Con el manejo de las propiedades es posible controlar el comportamiento de la aplicación. Las propiedades contienen información sobre la configuración del hardware.

Datos

Pueden ser datos provenientes de un sensor conectado a un subsistema de entrada analógica para ser almacenados en MATLAB o también pueden ser datos de salida de MATLAB a un actuador conectado a un subsistema de salida analógica.

Eventos

Un evento ocurre en un tiempo particular una vez que ciertas condiciones se hayan cumplido y se produzcan una o más acciones predeterminadas. Los eventos solo pueden generarse después de haber configurado las propiedades correspondientes. Una forma de utilizar los eventos puede ser el analizar o graficar datos una vez que se adquiere un número predeterminado de ellos.

Funciones M

Para ejecutar cualquier tarea con la aplicación de adquisición de datos, debe llamarse algunas funciones M. Entre otras cosas, estas funciones permiten:

- Crear dispositivos de objetos que proporcionan un camino de MATLAB al hardware y permite controlar el comportamiento de la aplicación.
- Capturar datos o sacar datos.
- Configurar las propiedades.
- Evaluar el estado y los recursos del hardware de adquisición.

El motor de adquisición de datos

Es una librería de enlace dinámico (.dll) en forma de archivo MEX que:

- Guarda los dispositivos de objetos y sus valores asociados de configuración que controlan la aplicación de adquisición de datos.
- Controla la sincronización de eventos.
- Controla el almacenaje de datos capturados o en espera de ser sacados.

Mientras el motor ejecuta estas tareas, puede usarse MATLAB para ejecutar otras tareas como el análisis de los datos adquiridos. En otras palabras, el motor y MATLAB son asíncronos.

El adaptador del manejador del hardware

El adaptador del manejador del hardware, o simplemente el manejador, es una interfaz entre el motor de adquisición de datos y el manejador de la tarjeta. El propósito principal del adaptador es pasar información entre MATLAB y la tarjeta adquisitora a través del manejador (driver) generalmente proporcionado por el fabricante [11].

Las tarjeta de sonido está presente en los ordenadores personales y facilita la entrada y salida de señales de audio bajo el control de un software. Este dispositivo se puede utilizar como un osciloscopio de señales hasta una frecuencia de 20 kHz.

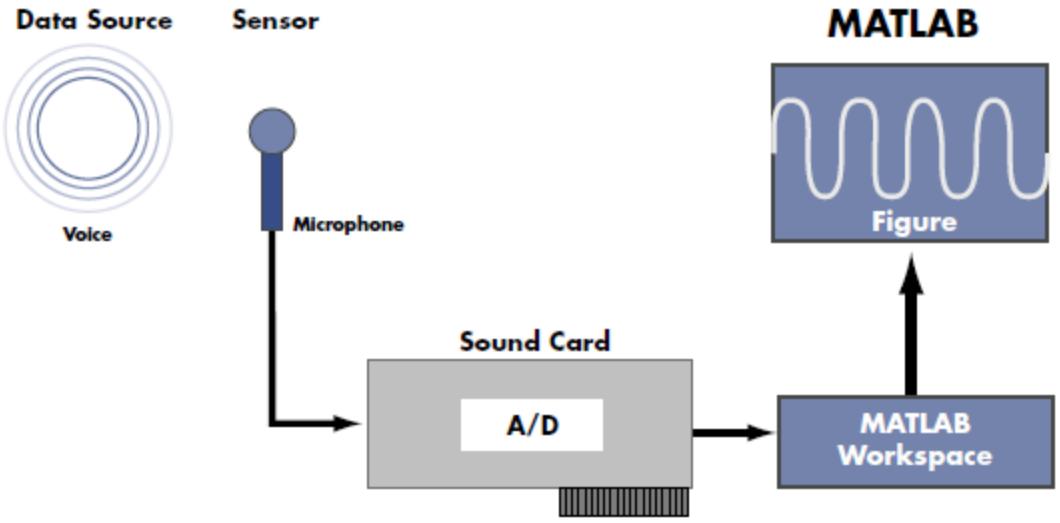


Figura 61. Esquema de la adquisición de datos y representación gráfica de la voz

A continuación se explica cómo la tarjeta de sonido se puede utilizar como un osciloscopio y un generador de señales mediante la adquisición de datos proporcionada por la toolbox correspondiente de MATLAB. La toolbox específica de adquisición de datos debe estar instalada para este propósito. Se usan dos canales para este fin, el canal izquierdo y el canal derecho. La señal de entrada se toma en la clavija de micrófono capturando datos mediante los dos canales y la salida se toma por la clavija de los auriculares. Véase *Figura 61*.

3.4.3.1 *El uso de la tarjeta de sonido como un osciloscopio*

En MATLAB existen dos aplicaciones GUI ya implementadas con las que se puede almacenar y representar en tiempo real los datos adquiridos mediante la tarjeta de sonido.

- Daqscope
- Softscope

Daqscope

"Daqscope" es un ámbito de aplicación simple que muestra la forma de onda de la señal de entrada de cualquiera de los canales y proporciona opciones para los límites máximo y mínimo de la onda. Este ámbito de aplicación puede ser iniciada al escribir la siguiente palabra en la ventana de comandos de MATLAB:

>>daqscope

La ventana "daqscope" es la siguiente:

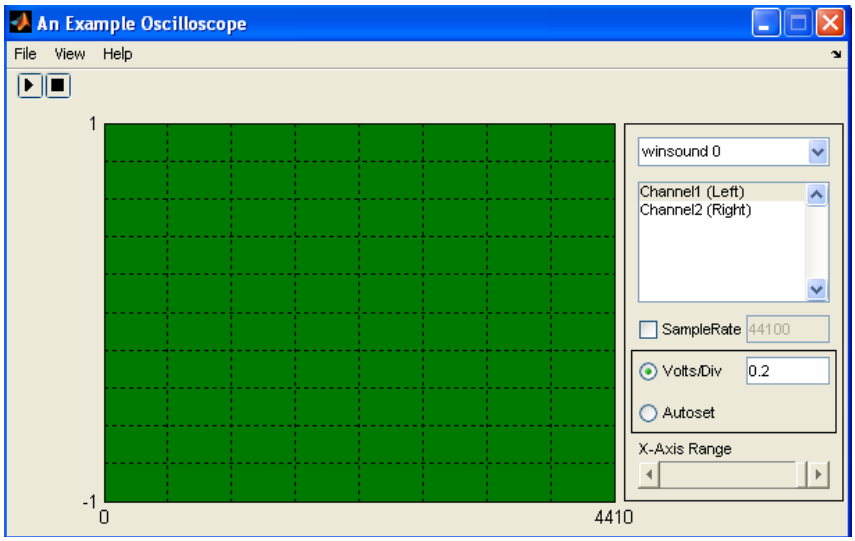


Figura 62. Ventana daqscope

Los botones situados en la esquina superior izquierda proporcionan el control del inicio de la detención de la adquisición de datos. El usuario puede seleccionar el canal deseado, puede modificar la frecuencia de muestreo y la escala. También ofrece la opción de configurar el auto que ajusta la amplitud y la escala del eje X de acuerdo a la señal que se prestan. Véase *Figura 62*.

La siguiente captura de pantalla muestra el osciloscopio en la acción de una onda sinusoidal de 1 kHz. Véase *Figura 63*.

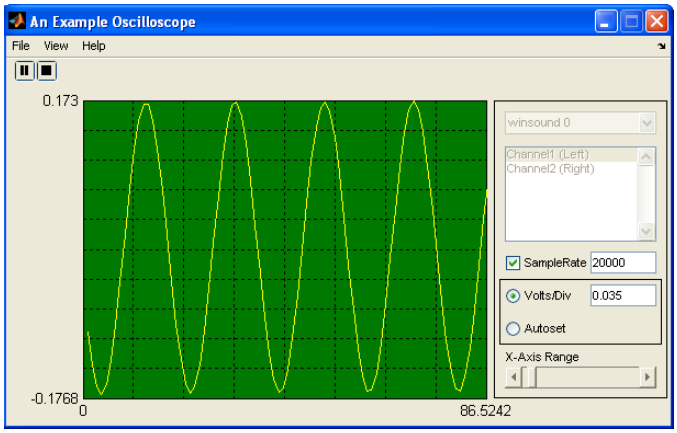


Figura 63. Representación de una onda sinusoidal en daqscope

Softscope

Este es otro ámbito de la aplicación que está implementada por MATLAB, que proporciona muchas más funciones que la "daqscope". "Softscope" puede ser iniciado por escribir la siguiente palabra en la ventana de comandos:

```
>>Softscope
```

Aparecerá la siguiente pantalla:

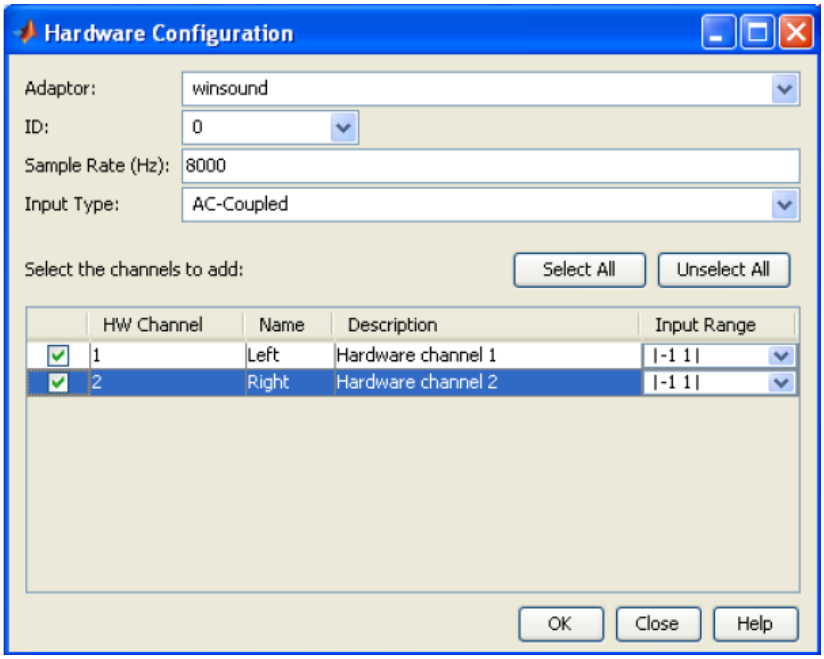


Figura 64. Ventana de configuración de softscope

Después de seleccionar las propiedades requeridas por ejemplo, frecuencia de muestreo o el canal deseado, la pantalla del osciloscopio que aparece es la siguiente:

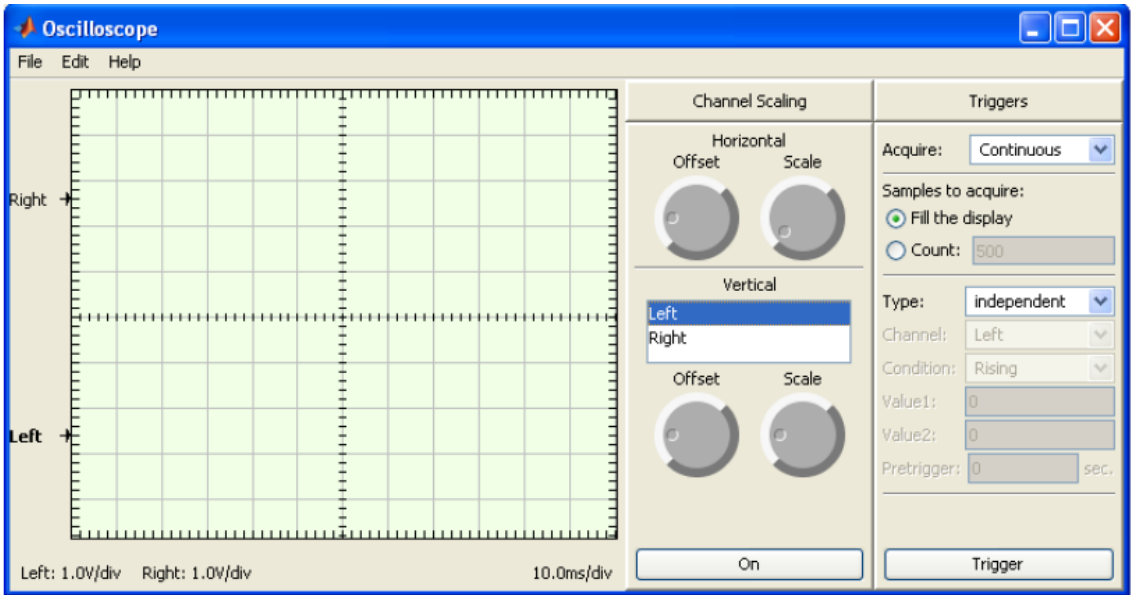


Figura 65. Ventana softscope

Esta ventana proporciona las propiedades de ajuste horizontal y vertical en la configuración del canal y diversas opciones de disparo. Véase *Figura 66*.

Después de aplicar una señal sinusoidal (320 Hz) en el canal izquierdo (canal 1), y pulsando el botón "Trigger", la señal siguiente es la mostrada en el osciloscopio:

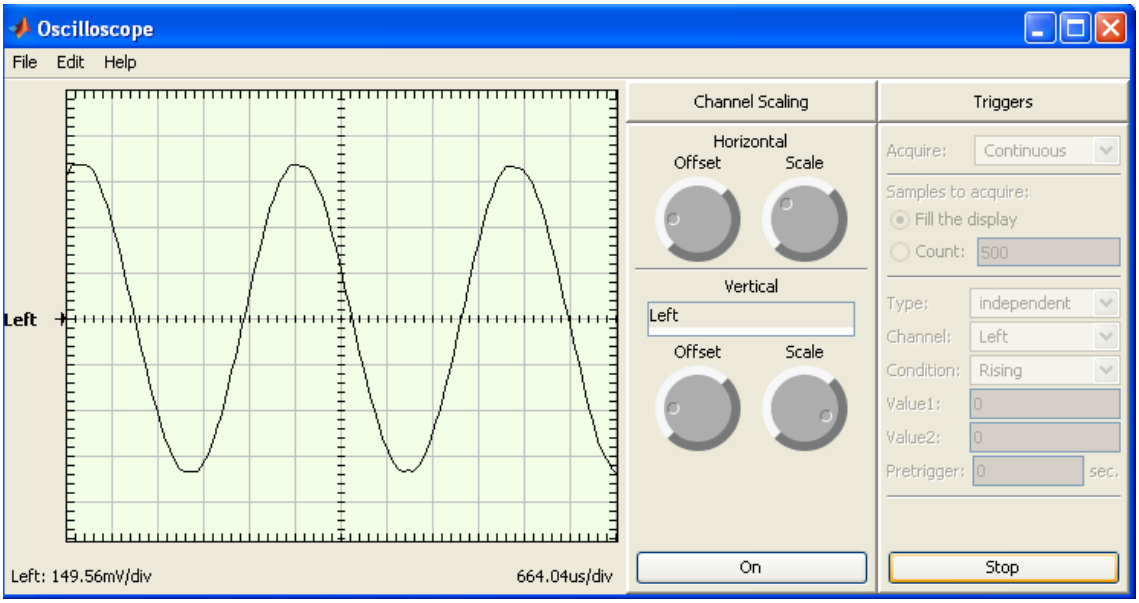


Figura 66. Representación de una onda sinusoidal en softscope

Después de aplicar una señal triangular (1,5 kHz) en el canal derecho y una onda cuadrada (1 kHz) en el canal izquierdo, *Figura 67*, la salida de campo se muestra como:

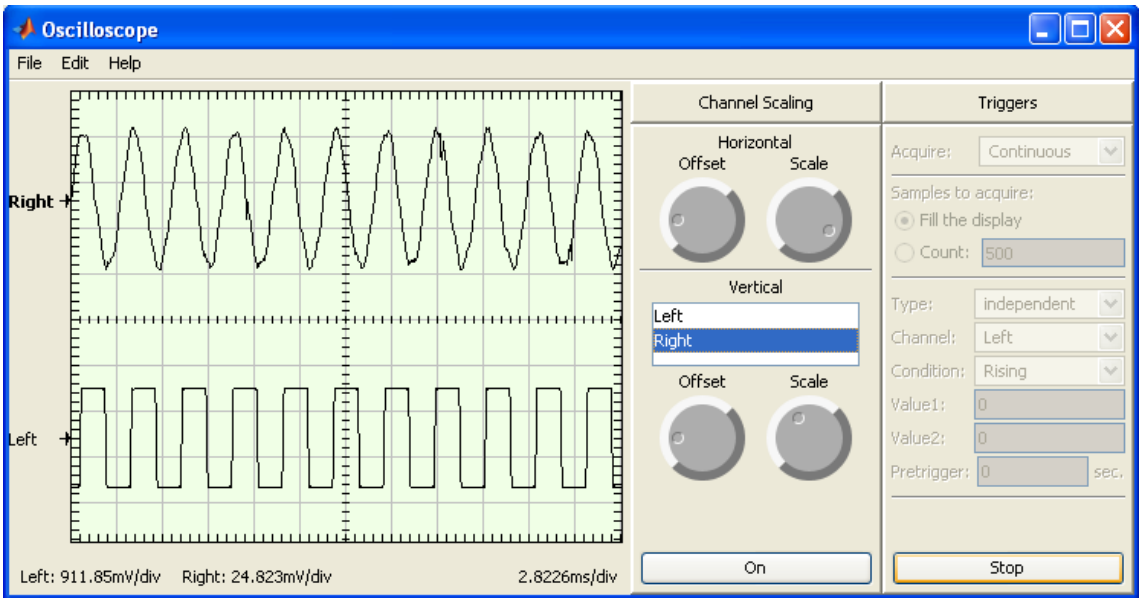


Figura 67. Representación de dos canales en softscope

"Softscope" proporciona las herramientas de medición siguientes que no están disponibles en "daqscope", *Figura 68,* que lo convierten en un ámbito útil para varias medidas:

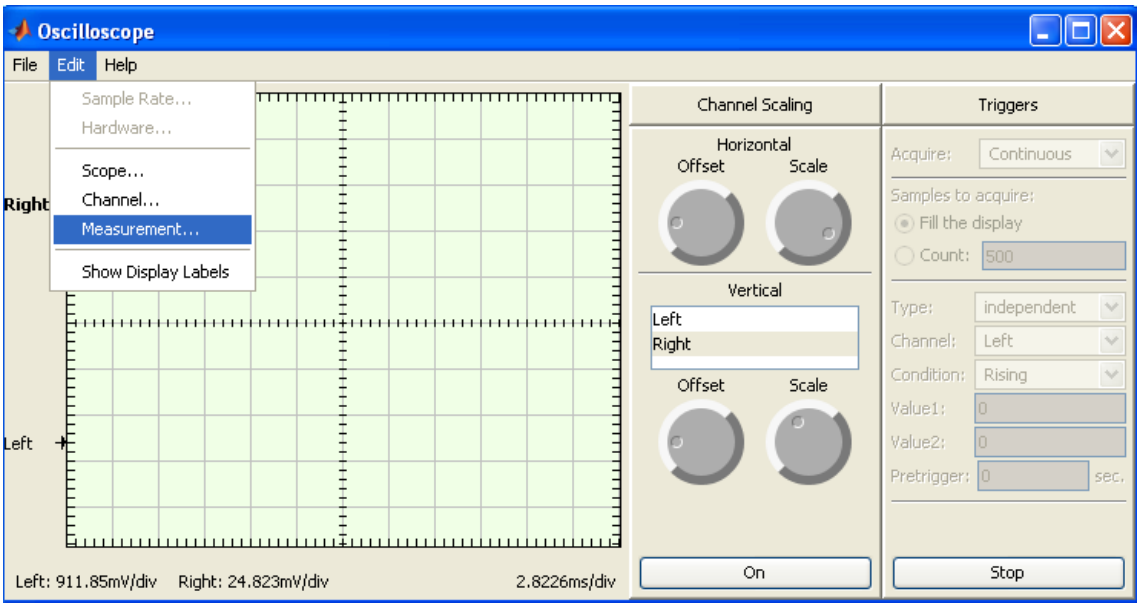


Figura 68. Menú de herramientas de la ventana softscope

Después de seleccionar las propiedades requeridas en el menú de las herramientas de medición, la siguiente ventana de softscope es:

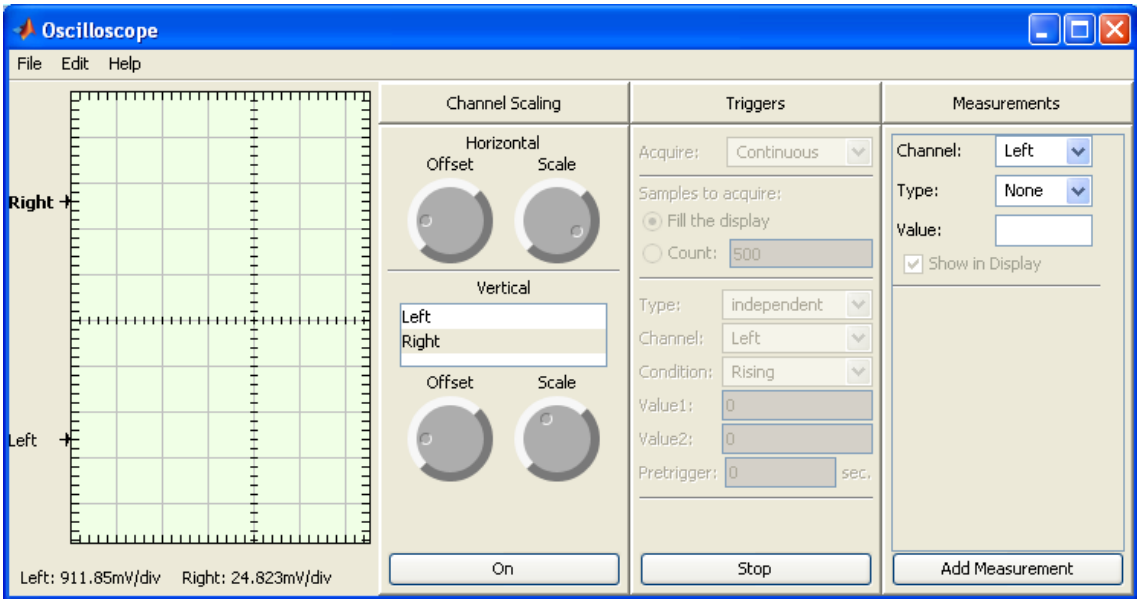


Figura 69. Ventana de softscope con el menú de herramientas integrado

Se selecciona en el menú el tipo de medición que se desee. La siguiente captura de pantalla, *Figura 70*, muestra la medición del valor medio de las dos señales antes mencionadas:

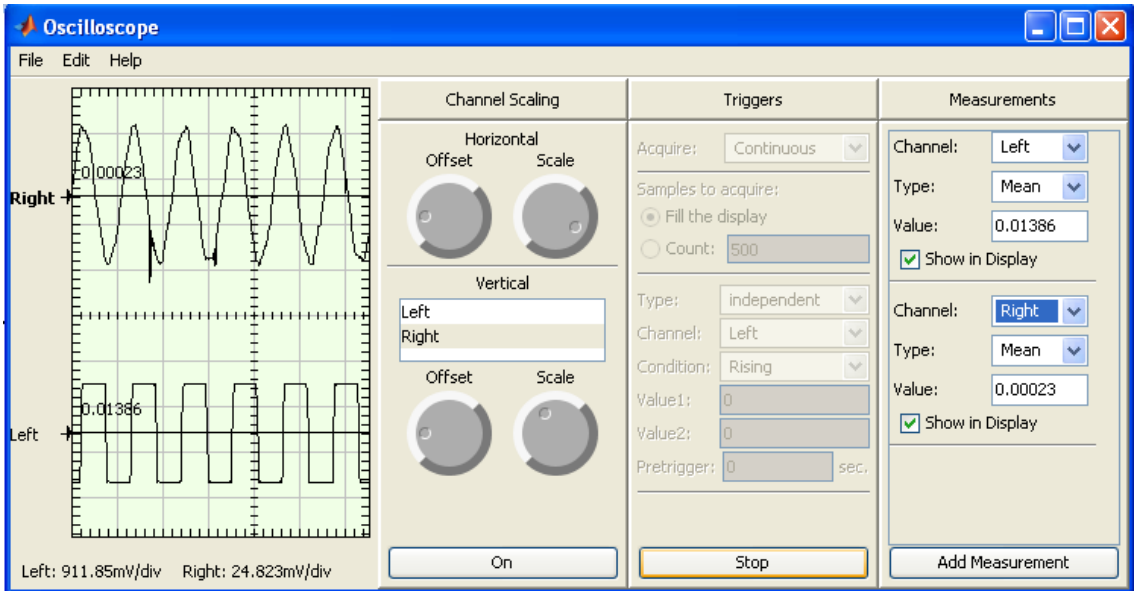


Figura 70. Representación de dos canales en la ventana de softscope expandida

Del mismo modo, se puede realizar otras mediciones diferentes. Softscope también ofrece la posibilidad de hacer una nueva medición según lo definido por el usuario, en forma de una función definida por el MATLAB.

3.4.3.2 *El uso de la tarjeta de sonido como un generador de funciones*

El generador de funciones DAQ pueden ser ejecutados por el siguiente comando de MATLAB:

>>daqfcngen

Esto abre la ventana del generador de funciones, como se muestra a continuación:

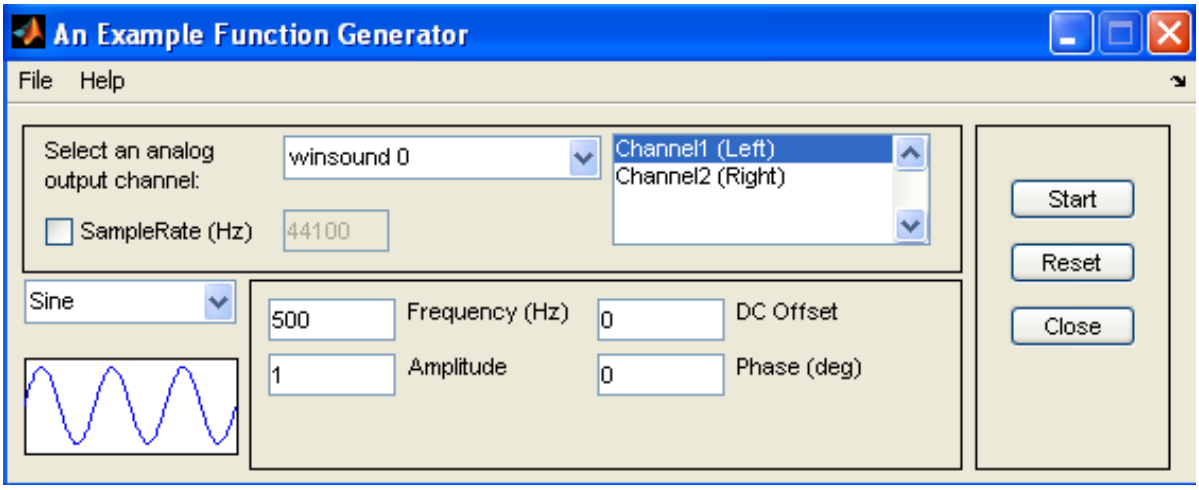


Figura 71. Ventana del generador de funciones

El generador de adquisición de datos ofrece la función de producción de diferentes formas de onda, *Figura 72*. Por ejemplo, sinusoidal, onda cuadrada, ruido y al azar. El usuario puede controlar la frecuencia, amplitud, offset de DC y la fase de la señal de salida (diferentes formas de onda tienen diferentes propiedades). Las formas de onda diferentes se pueden seleccionar de la siguiente manera:

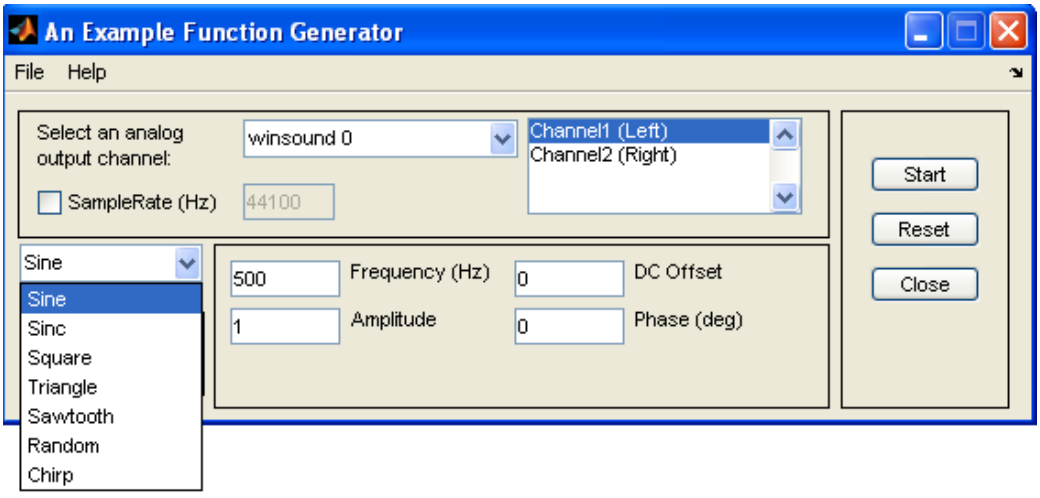


Figura 72. Tipo de señal a componer por el generador de funciones

A continuación, *Figura 73,* se presentan la tensión de salida "SawTooth" de este generador de funciones:

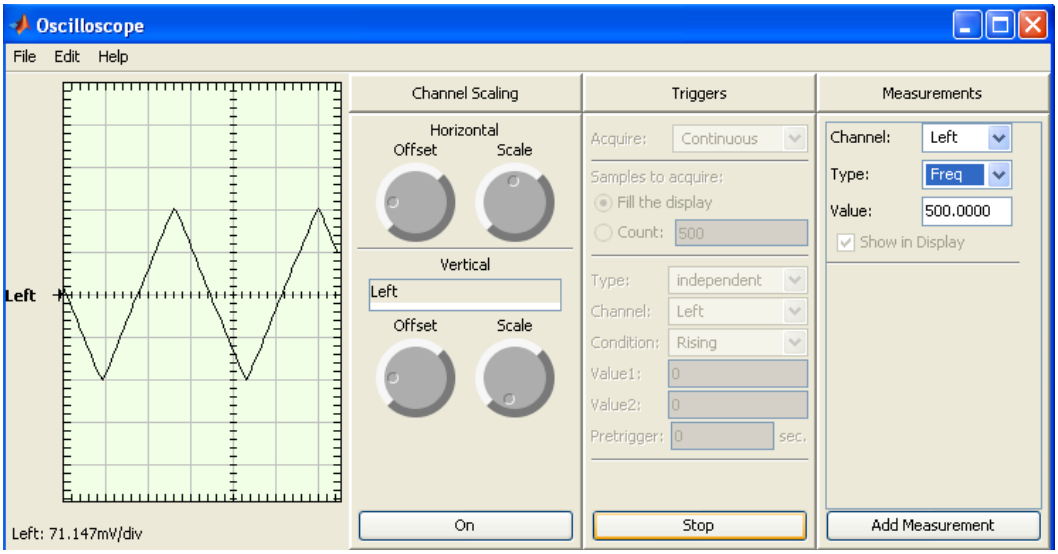


Figura 73. Representación de una señal triangular en el softscope

4. Desarrollo de la interfaz gráfica

4.1 Especificaciones de diseño

4.1.1 Requisitos del sistema

Las actuales herramientas que se utilizan para la adquisición de datos en las pruebas de fatiga y rotura de los rodamientos son máquinas costosas y complejas que no están optimizadas y son incapaces de realizar la adquisición de datos en tiempo real, hecho imprescindible para entender las características funcionales de cada rodamiento.

En vista de las necesidades ocasionadas, se requiere realizar una herramienta que sea completamente funcional y posea todas las características necesarias para llevar a cabo una precisa adquisición de datos en tiempo real e integrando todas las aplicaciones de tratamiento de señal en una sola interfaz gráfica.

Una de las características fundamentales es que la captación de datos en tiempo real tiene que tener un índice de repetitividad muy alto para poder ofrecer una información realista al usuario que monitoriza la interfaz gráfica y así comprobar el comportamiento ofrecido por los rodamientos de manera satisfactoria.

Los módulos que forman la herramienta de adquisición de datos son:

- Tarjeta de adquisición de datos KEITHLEY.
- Tarjeta de adquisición de datos integrada en el PC, micrófono.
- Modo simulación.

La herramienta diseñada para la adquisición de datos de rodamientos realiza simulaciones de forma virtual de situaciones reales cuando trabaja con datos anteriormente almacenados en otras sesiones para poder predecir las dificultades y errores que se tendrán a la hora de realizar la captación de datos reales, así como buscar las soluciones en caso de que se produzcan inconvenientes.

Por último, la interfaz gráfica deberá ser capaz de cargar, registrar y monitorizar en tiempo real los datos reales o simulados que recibe con precisión y exactitud suficiente para ofrecer una información de garantía.

4.1.2 Requisitos funcionales

La interfaz gráfica de usuario debe ser un sistema fundamentalmente visual para que el usuario que esté controlando la herramienta pueda tener en todo momento un conocimiento rápido y preciso de la situación actual en que se encuentra el comportamiento en fatiga y rotura de los ejes y rodamientos.

La información mostrada en la interfaz gráfica tiene que ser clara y concisa. Deben monitorizarse visiblemente tanto los valores numéricos más primordiales y transcendentes para el correcto conocimiento de la situación como las representaciones gráficas que indican visualmente la tendencia o dirección que adopta el rodamiento en su vida útil.

Se debe formar a los usuarios mediante el manejo de la interfaz gráfica en modo simulación utilizando datos anteriormente almacenados. Este tipo de plataforma permitirá una formación del usuario en un entorno casi real y ofrecen la posibilidad de simular cualquier situación que pueda darse en la práctica.

Las comunicaciones entre los diferentes módulos implicados en la adquisición de datos deben estar correctamente sincronizado para realizar automáticamente todo el flujo de intercambios de datos sin producirse ningún fallo de lectura o escritura de datos.

El tiempo de ejecución de cada software debe de estar totalmente depurado para que las variables almacenadas en el fichero lleguen rápidamente a la interfaz gráfica externa para que ésta pueda monitorizar las evoluciones dadas prácticamente en tiempo real.

4.2 Arquitectura del sistema

La herramienta diseñada es una arquitectura centralizada ya que la aplicación MATLAB instalada en un ordenador central es la autorizada de integrar bajo un mismo panel de control tres módulos bien diferenciados encargados de obtener, procesar, representar y transmitir los intercambios de datos necesarios para realizar una adquisición de datos lo más exacto y preciso posible de los rodamientos respecto de sus valores reales.

En primer lugar, se encuentra el módulo de la tarjeta de adquisición de datos KEITHLEY, encargada de obtener todas las señales analógicas y digitales reales comprendidas en un rango de 1 a 8 canales en tiempo real. La tarjeta KEITHLEY se conecta al ordenador mediante USB y se instalan los respectivos "drivers" ofrecidos por el fabricante para el correcto funcionamiento de ésta en el sistema operativo establecido. De esta forma, se procesan, modifican y monitorizan los datos reales adquiridos.

En segundo lugar, se encuentra el módulo de la tarjeta de sonido del micrófono que está integrada en el propio ordenador. No necesita ninguna instalación de "drivers" o configuración determinada, lo que permite la adquisición de datos únicamente con la programación software. Al habilitar el micrófono para la captación de datos permite la captura de dos canales independientes simultáneamente de cualquier dispositivo que se conecte a la clavija del propio micrófono. Por ejemplo, acelerómetros.

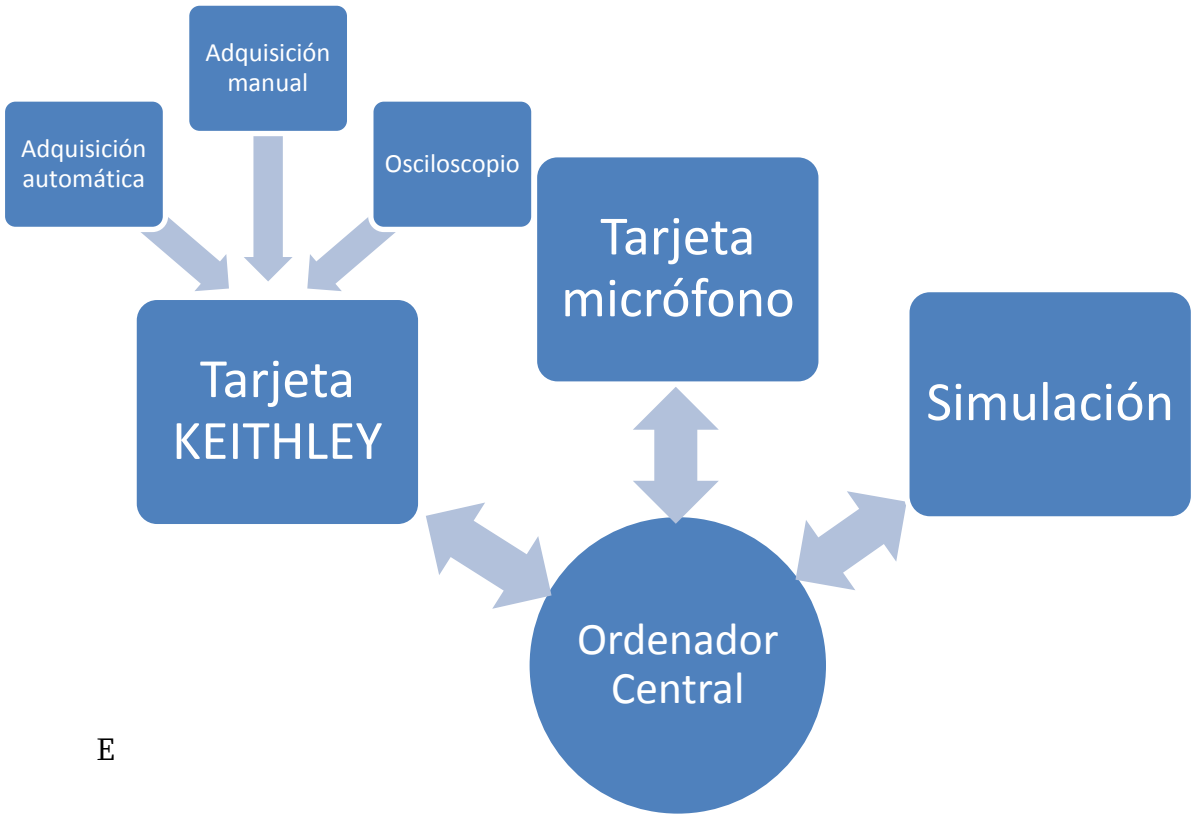
En tercer lugar, se encuentra el módulo de simulación, encargado de cargar todos los datos obtenidos en una anterior sesión para simular la adquisición de datos reales. Este modo de operación está diseñado únicamente a nivel de software por lo que dispone de una gran capacidad funcional y además permite ampliaciones y mejoras continuas.

Por último, se encuentra el ordenador central desde donde se controlan todos los flujos de información entre los tres módulos anteriormente citados. Mediante la instalación de la aplicación MATLAB y su correspondiente programación software, se diseña el panel de mandos central *"Btool"* desde donde se dirigen todas las acciones a realizar por el usuario. El intercambio de información por parte de los tres módulos debe de estar rigurosamente sistematizado para que el sistema no se colapse por la gran cantidad de datos entrantes y salientes durante toda la toma de datos.

Otra función esencial del módulo central es la representación gráfica de los datos reales y simulados. Del mismo modo, debe quedar representada y almacenada toda la información paramétrica que necesita el usuario, ofrecer al usuario la opción de representación real o simulada y además poder alternar y comparar entre ambas.

4.2.1 Interfase: PC con KEITHLEY, micrófono y simulación

En este apartado se va a explicar detalladamente el flujo de operación y datos que mantiene el ordenador central con sus respectivos módulos de trabajo que son la tarjeta KEITHLEY, la tarjeta del micrófono y el modo simulación. Véase *Figura 74*.



E

Figura 74. Flujo de operación

En primer lugar partimos de la base en que los tres módulos y el ordenador central se encuentran inicializados y preparados para el comienzo correcto de la herramienta.

A continuación, se inicializa la aplicación software MATLAB en el ordenador central y se ejecuta el programa principal *"Btool"* que lanza el panel de mandos desde donde el usuario tiene la posibilidad de elegir el modo de captación o representación de datos. Independientemente del modo que elija se accederá a un panel de control secundario llamado *"Allchannels"* desde donde el usuario podrá realizar un tratamiento de la señal modificando y adaptando la información visual a sus necesidades.

Si el usuario elige la opción de captación de datos mediante la tarjeta KEITHLEY, *Figura 75*, se inicializarán las variables y parámetros correspondientes para su correcto funcionamiento y se accederá al panel de control secundario anteriormente citado desde donde se da opción al usuario a trabajar con tres modos de adquisición de datos diferentes. Los modos de adquisición de datos son:

- **Adquisición de datos automática:** la tarjeta KEITHLEY recopila información de las señales entrantes en su rango de 1 a 8 canales durante un tiempo establecido por usuario y a continuación refleja por pantalla su representación en función de los valores de voltaje en el tiempo.

- **Adquisición de datos manual:** la tarjeta KEITHLEY monitoriza la información de un único canal seleccionado en tiempo real por pantalla ofreciendo la posibilidad al usuario de empezar la adquisición de datos cuando desee, es decir, en el momento que detecte que la señal proveniente de los esfuerzos sometidos a fatiga de los rodamientos están estabilizados, el usuario daría la orden de comenzar a capturar datos ya que éstos son los que verdaderamente interesan para el posterior análisis. A continuación representa por pantalla las dos señales captadas, es decir, la señal que se adquiere desde el principio del lanzamiento de la aplicación y la señal que se adquiere desde la ordenanza del usuario. De esta manera éste puede comparar visualmente la evolución de ambas señales en el tiempo y tener la posibilidad de trabajar con los datos almacenados de cualquiera de ellas a su conveniencia.

- **Osciloscopio:** este modo de adquisición es muy semejante al anterior pero con la diferencia que cuando la tarjeta KEITHLEY adquiere información, ésta se monitoriza en tiempo real en una aplicación software propia de MATLAB que simula un osciloscopio analógico permitiendo una mayor funcionalidad. De este modo, el usuario puede trabajar con la señal desde la propia aplicación lanzada teniendo opción a realizar tratamientos de señal, escalamiento, reducción del offset… antes de proceder a la adquisición de los datos definitiva.

Figura 75. Tarjeta de adquisición de datos KEITHLEY

Si el usuario elige la opción de adquisición de datos mediante la tarjeta del micrófono, se inicializan por defecto las variables y parámetros correspondientes para poder trabajar correctamente en el panel secundario *"Allchannels"* y proceder a la captación de datos por medio de la tarjeta de sonido integrada en el ordenador. La adquisición de datos corresponde a dos canales pertenecientes al audio de la izquierda y derecha del micrófono integrado en el ordenador. Sin embargo, podemos conectar cualquier dispositivo a la clavija física del micrófono y disponer de dos canales independientes de adquisición de datos sin necesidad de necesitar la tarjeta de adquisición KEITHLEY. Esto resulta de gran utilidad ya que podemos utilizar la herramienta en cualquier ordenador y ser capaces de capturar por la clavija del micrófono dos canales de datos independientes. Véase *Figura 76*.

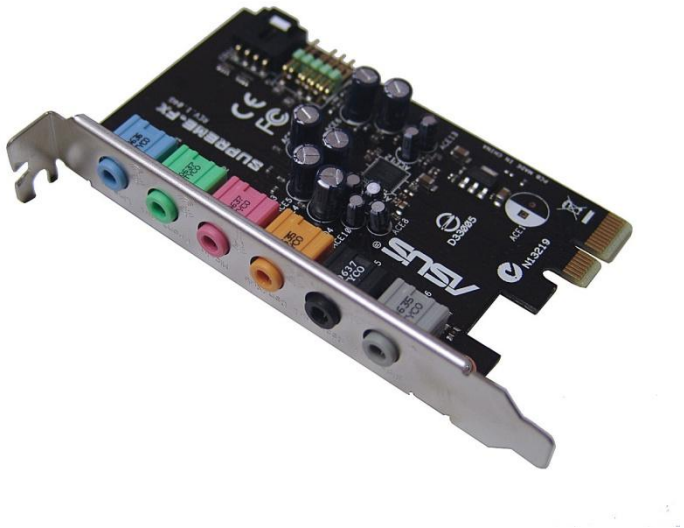


Figura 76. Tarjeta de sonido

Por último, se encuentra el modo simulación en el que no adquirimos información de ningún dispositivo interno o externo. Simplemente es un modo que permite representar adquisiciones de datos anteriormente almacenados para trabajar con ellos o representar ondas sinusoidales perfectas con una escala de tiempos diferente. Esto es beneficioso tanto para desarrollar experimentos de los usuarios como para ayudar a la formación de éstos.

4.2.2 Procesado de información

Para el almacenamiento y control de todas las variables y los respectivos flujos de información dentro de la propia aplicación software de MATLAB existen unos manejadores o identificadores llamados *"handles"* que están destinados a almacenar y estructurar jerárquicamente dichas variables de manera que sean visibles y disponibles para el resto de las funciones del fichero.

Cada uno de los objetos de MATLAB tiene un identificador único a los que se les llamará *"handle"* o *"id"*. Algunos gráficos tienen muchos objetos, en cuyo caso tienen múltiples *"handles"*. El objeto raíz (pantalla) es siempre único y su identificador siempre es cero. El identificador de las ventanas siempre es un entero que aparece en la barra de nombre de dicha ventana. Los identificadores de otros elementos gráficos son números float.

En MATLAB puede haber múltiples ventanas abiertas pero solo una esta activa. Cada una de estas ventanas puede tener ejes abiertos pero solo se dibuja en los ejes activos. Los identificadores de la ventana activa de los ejes activos y del objeto activo se pueden obtener con los siguientes comandos:

- **gcf(get current figure):** Devuelve el entero que es el handle de ventana activa.
- **gca(get currant axis):** Devuelve handle de los ejes activos.
- **gco(get current object):** Devuelve handle del objeto activo.
- **delete handle:** borra el objeto correspondiente y todos sus hijos.

Sin embargo, para intercambiar dichas variables estructuradas de un fichero a otro se necesitan unas instrucciones dentro de la propia aplicación llamadas *"varargout"* y *"varargin"*. Estas son funciones ya implementadas cuya misión es transferir los argumentos de entrada y de salida de unas funciones a otras entre diferentes ficheros pertenecientes a interfaces gráficas *"GUI"*.

No obstante, hay algunas situaciones en que la programación del software no permite realizar los intercambios con dichas instrucciones propias citadas anteriormente y se necesita recurrir a la utilización de variables globales. Éstas son visibles y accesibles para cualquier función o fichero donde son declaradas, sin necesidad de recurrir a los argumentos de entrada y salida de las funciones. Sin embargo, la programación con variables globales no favorece la optimización del software y por tanto el tiempo de respuesta se amplía debido a la mayor carga de trabajo que implican dichas variables globales.

Por último, se utilizan las variables locales dentro de las funciones de un fichero para aquellas variables que no son importantes y no se necesitan intercambiarlas a otras funciones o ficheros. Solamente son visibles dentro de la propia función donde son declaradas y por tanto, no ocupan casi memoria en el disco duro y son una forma de programación software muy efectiva.

4.3 Implementación del software de la interfaz gráfica

4.3.1 Descripción del sistema

El sistema de la interfaz gráfica de usuario (GUI) está formado por dieciséis archivos o funciones **.m* y se pueden llegar a generar multitud de archivos de intercambio de variables **.mat* y archivos de texto donde se almacenan los datos adquiridos en cada sesión. Todos estos archivos se encuentran almacenados dentro de una carpeta de trabajo del programa MATLAB. Véase listado de funciones en Anexo B.

Para realizar el intercambio de variables se ha recurrido a la creación de una estructura propia dentro de los propios manejadores que ofrece MATLAB. Es una estructura que se encuentra en un segundo nivel, es decir, es una estructura que se encuentra dentro de otra. Esto es debido a que las diversas funciones GUI que se utilizan comparten las mismas variables con los mismos nombres en el primer nivel de la estructura. Por ello, se sobrescribirían las variables propias que se generan en cada función GUI al intercambiarse a otra función GUI ya que comparten los mismos nombres por defecto.

Por tanto, al realizar el intercambio de todas las variables almacenadas en los *"handles"* de un fichero GUI a otro, las variables del fichero 1 almacenadas en *"handles.XXX"* se borran porque se sobrescriben las *"handles.XXX"* del fichero 2 que tienen el mismo nombre. Sin embargo, las variables almacenadas en *"handles.datos.XXX"* son independientes de las variables propias de MATLAB y son comunes a todos los ficheros modificándolas convenientemente según en la parte del programa que no encontremos.

A continuación se muestra un ejemplo gráfico de la estructura de almacenamiento que tienen las variables para así poder entender cómo se realizan los flujos e intercambios de datos entre las diferentes funciones o ficheros GUI. Véase *Figura 77*.

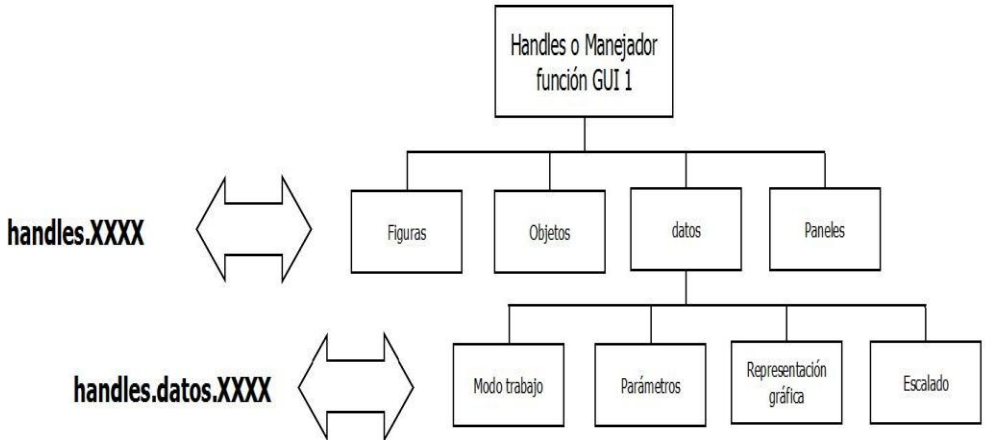


Figura 77. Estructura de almacenamiento de los Handles

Btool.m

El sistema de monitorización de la adquisición de datos comienza cuando se ejecuta la función principal *"Btool.m"* desde la ventana de comandos del propio programa o pulsando el botón *"Run"* de la propia función. Véase *Figura 78*.

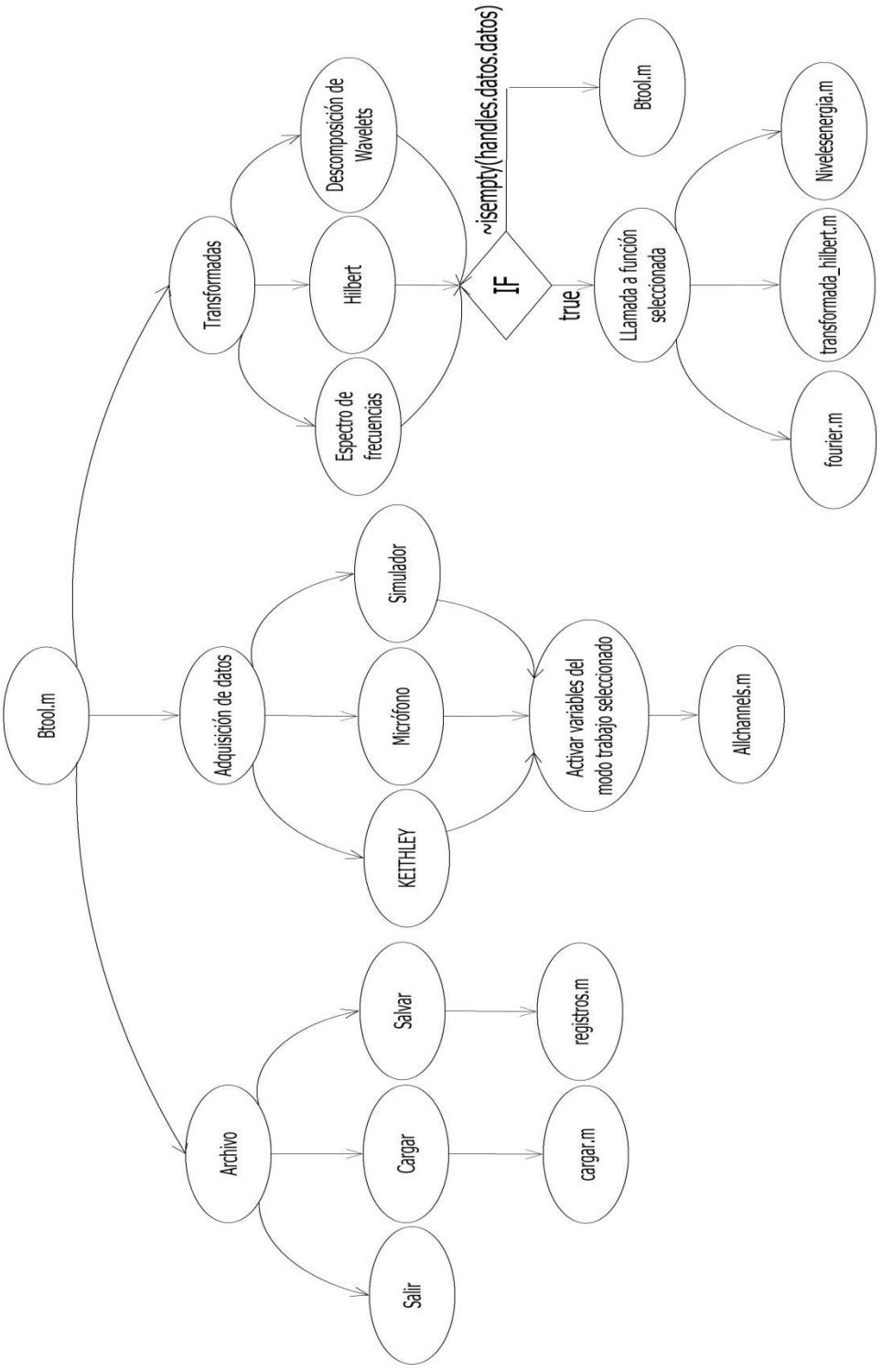


Figura 78. Diagrama de flujo la función "Btool.m"

Esta función ejecuta al iniciarse una serie de condiciones iniciales imprescindibles para el correcto funcionamiento de la aplicación durante la sesión de trabajo. Declara variables con o sin valor preestablecido relacionadas con el modo de trabajo en el que se encuentra, valores de parámetros, representaciones gráficas, configuraciones iniciales, escalados e incluso fecha y hora de la sesión iniciada. Estas condiciones primarias son esenciales para poder controlar en todo momento el flujo de información y garantizan que no haya errores en los diferentes modos de trabajo, no produciéndose duplicados de ellas y de esta manera se pueda tener un control total de todas las variables al intercambiarse de un fichero a otro.

Una vez establecidas dichas condiciones de inicio, se lanza el panel de control de la interfaz gráfica principal donde se dispone de varios menús contextuales que permiten al usuario elegir entre los diferentes modos de trabajo o realizar la carga de una sesión de trabajo anterior. Dependiendo de la opción elegida el flujo que sigue el software es totalmente distinto. Véase *Figura 79*.

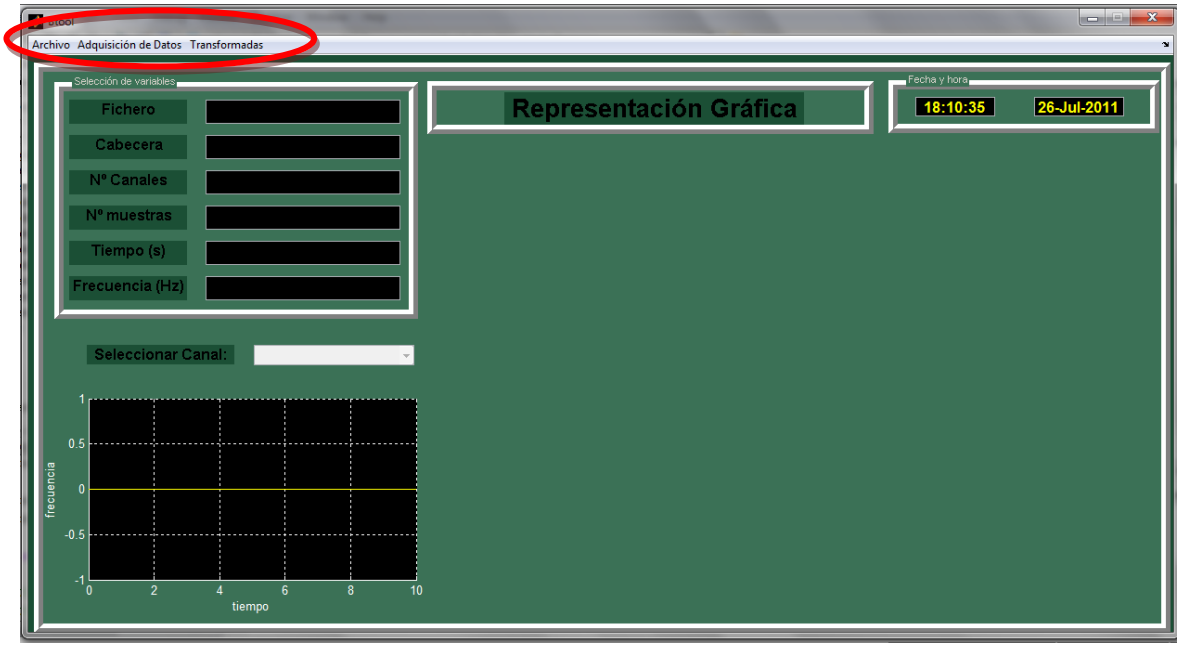


Figura 79. Panel de control de la interfaz gráfica "Btool"

Esta es la ventana principal de la interfaz gráfica desde donde el operario es capaz de coordinar los diferentes modos de trabajo desde el menú *"Adquisición de Datos"* y de seleccionar las diferentes opciones para realizar el tratamiento de señal mediante el menú *"Transformadas"*.

A continuación, van a describirse los diferentes ficheros *".m"* que intervienen en la programación del fichero *"Btool"* de la interfaz gráfica y sus respectivos diagramas de flujo para explicar de forma visual e intuitiva el desarrollo de la programación software de la herramienta diseñada.

cargar.m

Este fichero corresponde a una función convencional ya que no dispone del respectivo *".fig"*, es decir, la figura o GUI asociada a dicho fichero. Por tanto, una vez que el usuario ha pulsado el botón cargar en el menú contextual de la interfaz de *"Btool"* se ejecuta directamente el código asociado a dicha función y no se proyecta una nueva interfaz diferente.

La función *"cargar.m"* consiste fundamentalmente en seleccionar un archivo guardado previamente en otra sesión para poder cargarlo en la interfaz gráfica de *"Btool"* y poder trabajar con él. Si no existe ningún fichero para realizar la carga o el usuario no selecciona ninguno, la interfaz regresa de nuevo al panel de mando *"Btool.m"* y se mantiene a la espera de recibir nuevas instrucciones.

Los archivos guardados previamente se almacenan con un formato y estructura predeterminada para facilitar la lectura de las variables necesarias como la frecuencia y el número de muestras, el nombre y número de los canales seleccionados y por último los datos almacenados. Véase *Figura 80*.

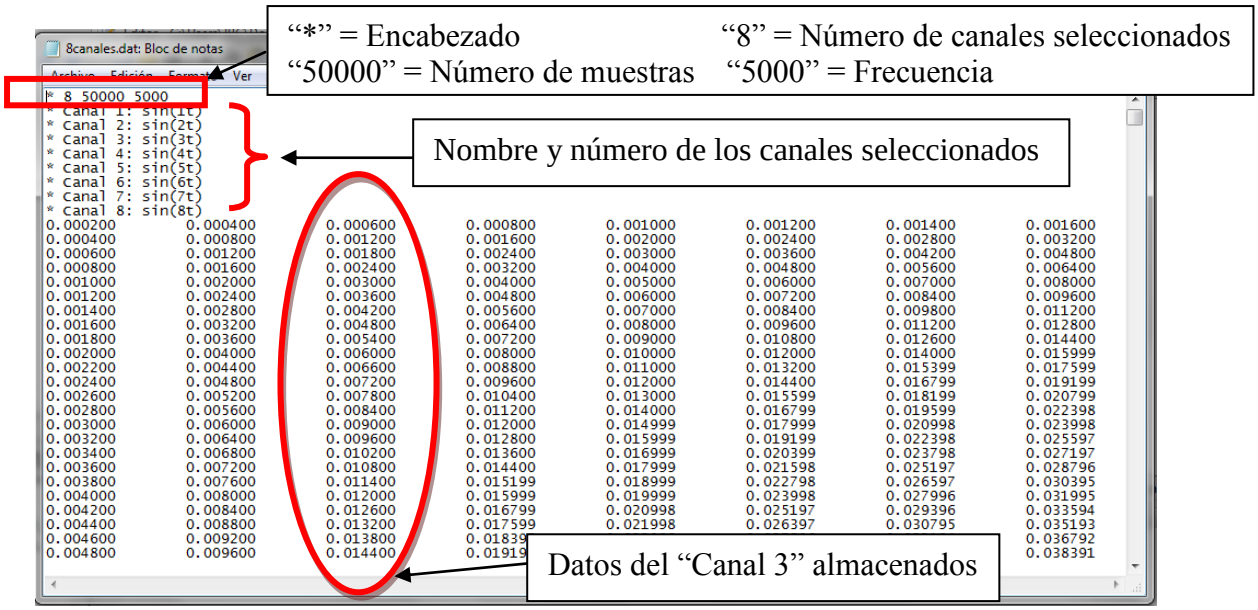


Figura 80. Archivo de almacenamiento de datos de la señales adquiridas

Una vez que se han capturados todos los datos necesarios de la señal del archivo de texto *".txt"*, el programa ejecuta las funciones *"parametros.m"*, *"posicion_ejes.m"* y *"representacion.m"*. Estas funciones son las encargadas de calcular todos los parámetros y coordenadas necesarias para poder llevar a cabo la representación gráfica de todos los datos adquiridos y mostrar en la interfaz al usuario la información paramétrica más relevante para el correcto control y dirección del panel de control.

A continuación, se muestra el diagrama de flujo de la función "cargar.m". Dependiendo de si los fichero *".txt"* tienen cabecera o no seguirán un camino u otro pero compartiendo las funciones básicas para llevar a cabo la representación gráfica. Véase *Figura 81*.

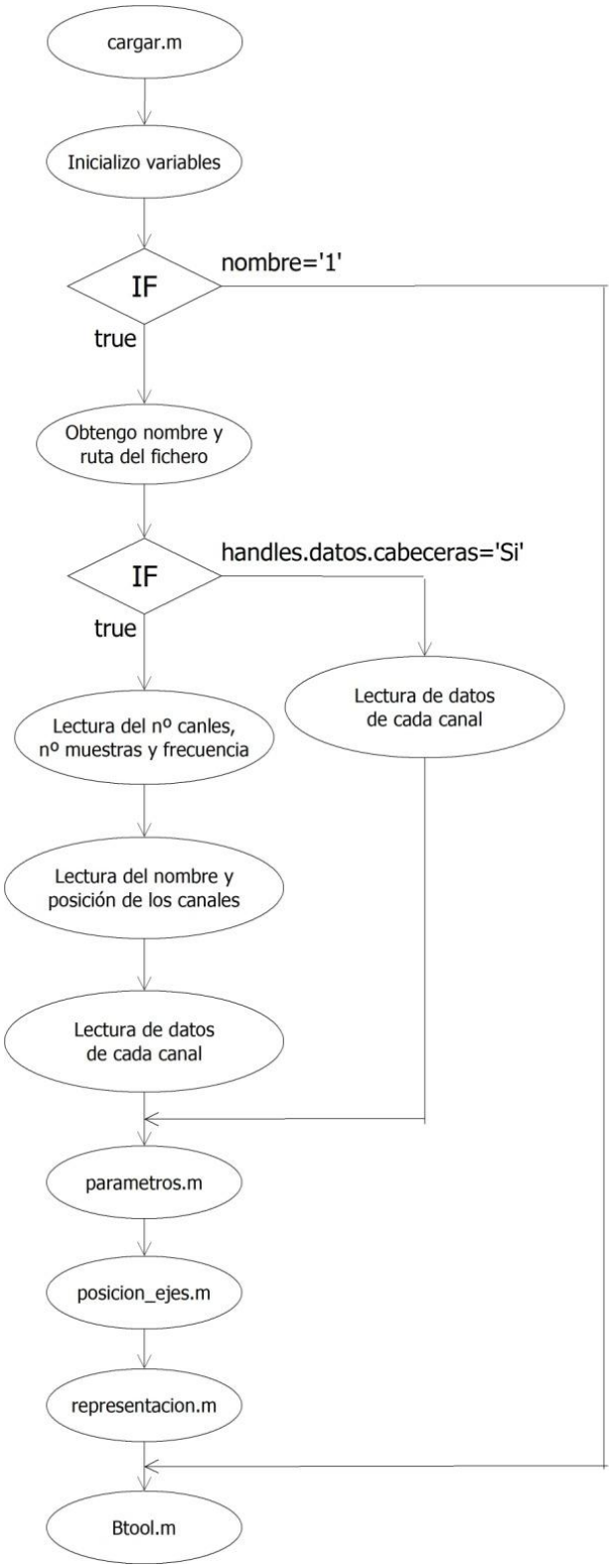


Figura 81. Diagrama de flujo de la función "cargar.m"

registros.m

La función *"registros.m"* consiste básicamente en almacenar en un fichero de texto *".txt"* los datos más relevantes después de realizar una representación gráfica. Por tanto, almacenamos todos aquellos parámetros detallados en la función *"cargar.m"*.

Si el programa detecta que no hay datos posibles que almacenar regresa de nuevo a la función *"Btool.m"* sin realizar ningún almacenamiento de datos. Por el contrario, si al pulsar el botón *"Salvar"* detecta datos que han sido previamente representados gráficamente, los almacena en un archivo de texto siguiendo una estructura predeterminada.

A continuación, se muestra el diagrama de flujo de operación de la función *"registros.m"*. Véase *Figura 82*.

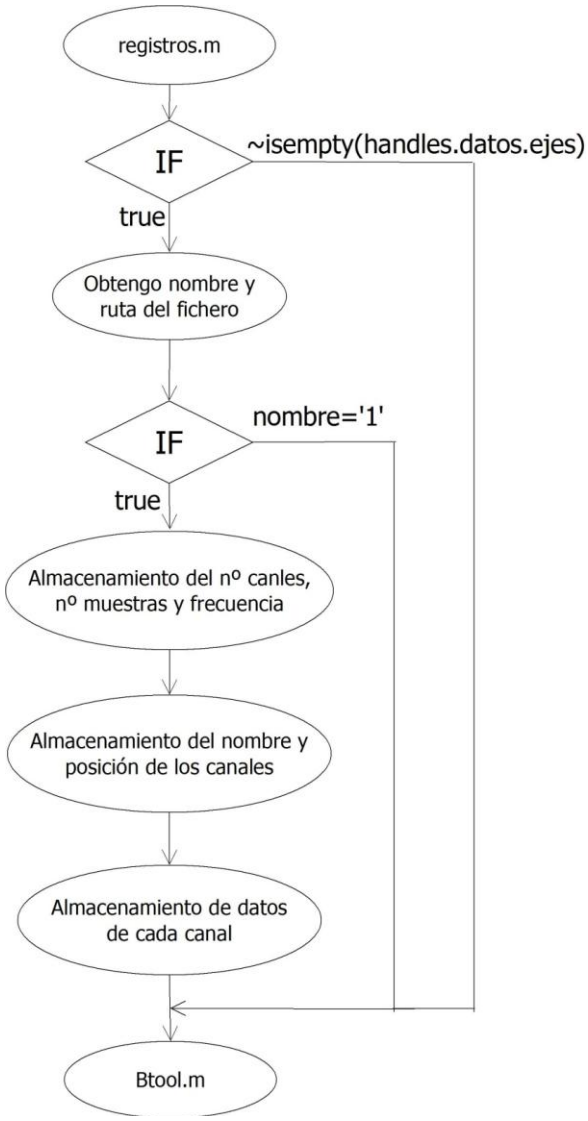


Figura 82. Diagrama de flujo de la función "registros.m"

Acto seguido, se van a explicar las opciones pertenecientes al menú contextual *"Transformadas"* que se utilizan en el panel principal *"Btool"* para realizar el tratamiento de señal de los datos capturados. La transformación que se realiza puede ser de tres tipos: Espectro de frecuencia, Transformada de Hilbert y Cálculo de los Niveles de Energía. Estas transformaciones se realizan de un solo canal, es decir, únicamente del canal seleccionado en el *"Pop up Menu"* de la interfaz gráfica. Véase *Figura 83*.

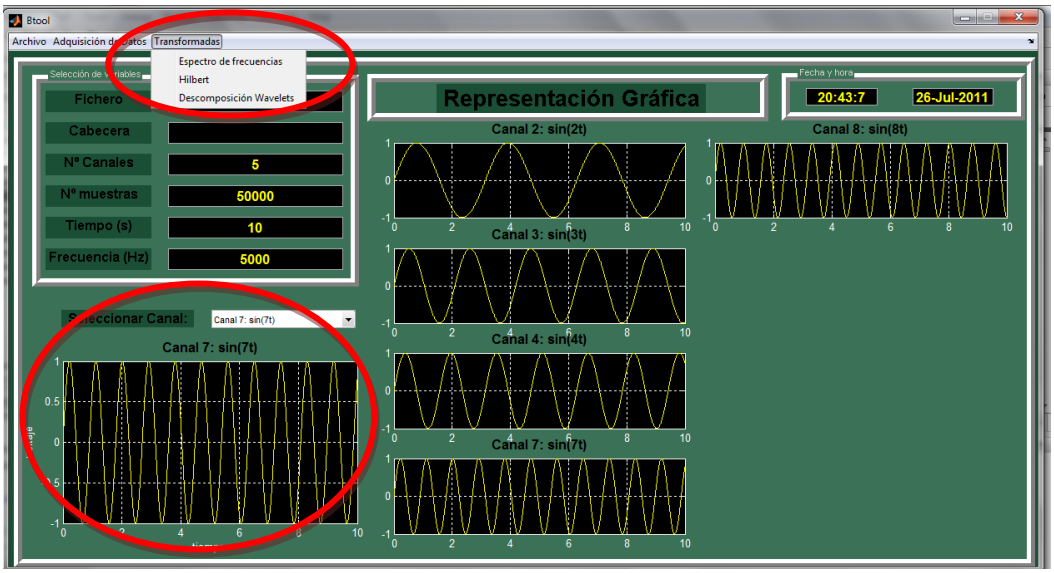


Figura 83. Menú transformadas del panel de control de la interfaz "Btool"

espectro.m

La función *"espectro.m"* tiene asociada un *".fig"*, por lo que dicha función dispone de una interfaz gráfica propia. Véase *Figura 84*.

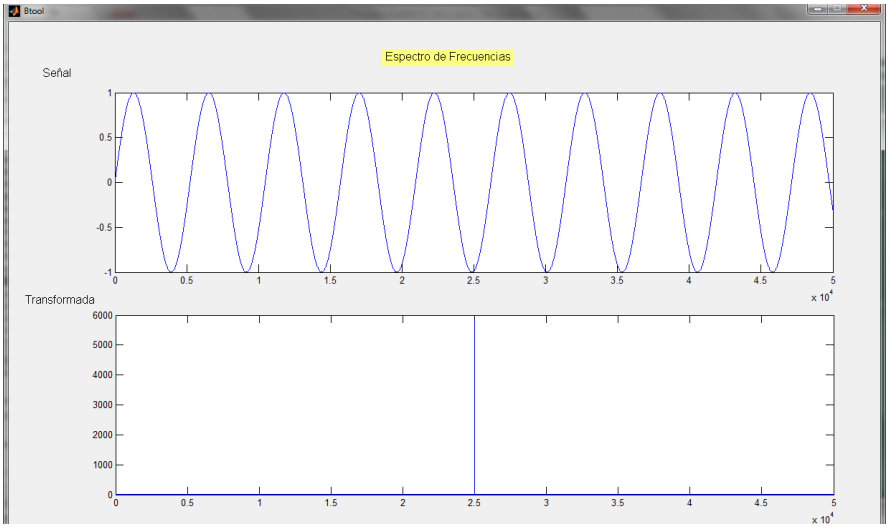


Figura 84. Interfaz gráfica de la función "espectro.m"

Esta función realiza la transformada de Fourier a la señal seleccionada en el *"Pop up Menu"* de la interfaz principal de *"Btool"*. Su único objetivo es la visualización tanto de la señal original capturada como de su transformada de Fourier por lo que el usuario no dispone de funcionalidad o control sobre la interfaz.

A continuación, se muestra el diagrama de flujo de la función *"espectro.m"* donde se puede apreciar la condición que si no detecta datos almacenados no calcula la transformada respectiva y accede directamente a la interfaz principal *"Btool"*. Véase *Figura 85*.

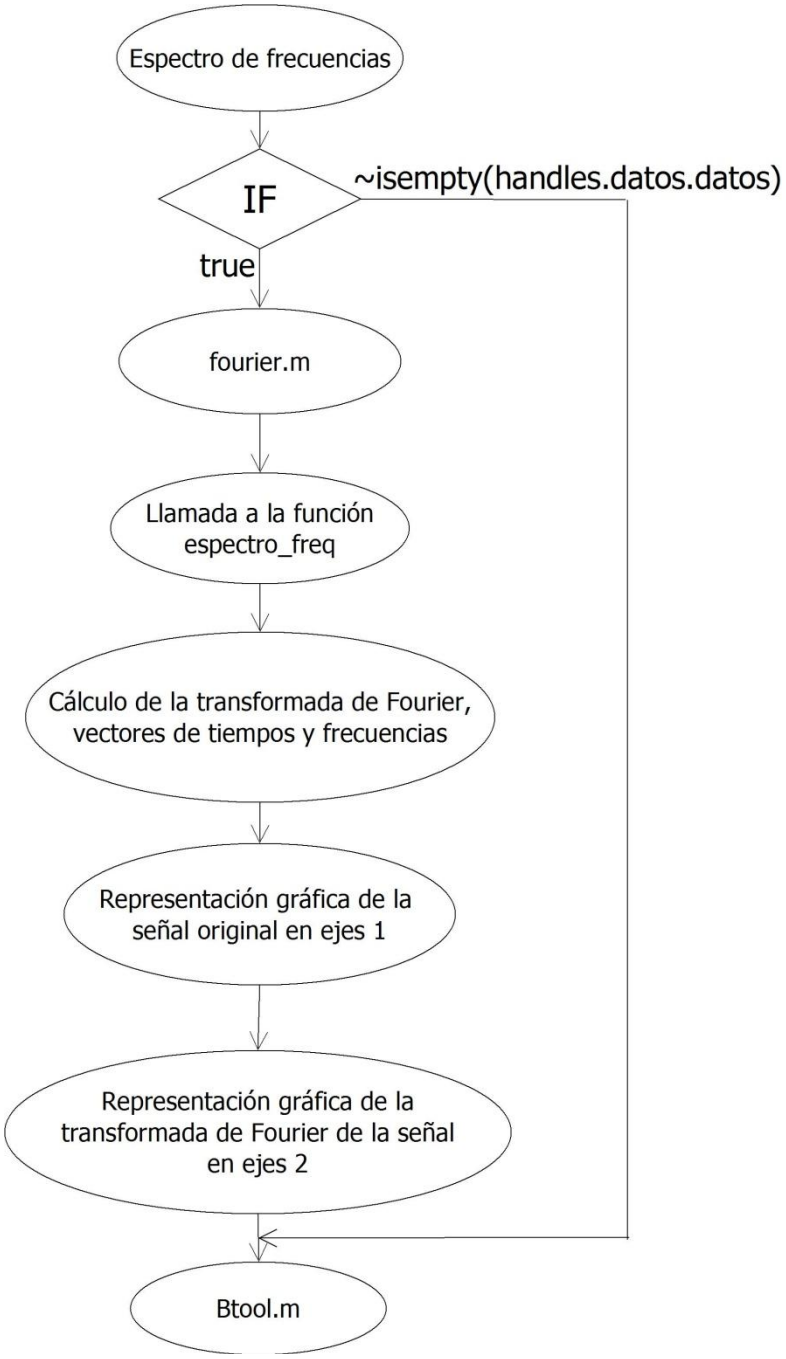


Figura 85. Diagrama de flujo de la función "espectro.m"

transformada_hilbert.m

La función *"transformada_hilbert.m"* tiene asociada un *".fig"*, por lo que dicha función dispone de una interfaz gráfica propia. Esta interfaz es igual que la interfaz gráfica de la función *"espectro.m"*. Véase *Figura 85*.

Por tanto, la función *"transformada_hilbert.m"* es muy semejante a la función *"espectro.m"* ya que tiene la misma estructura, flujo de operación e interfaz gráfica.

Sin embargo sus resultados y objetivos son diferentes ya que su finalidad es distinta. Son dos tratamientos de señal independientes entre sí. A continuación, se muestra su diagrama de flujo. Véase *Figura 86*.

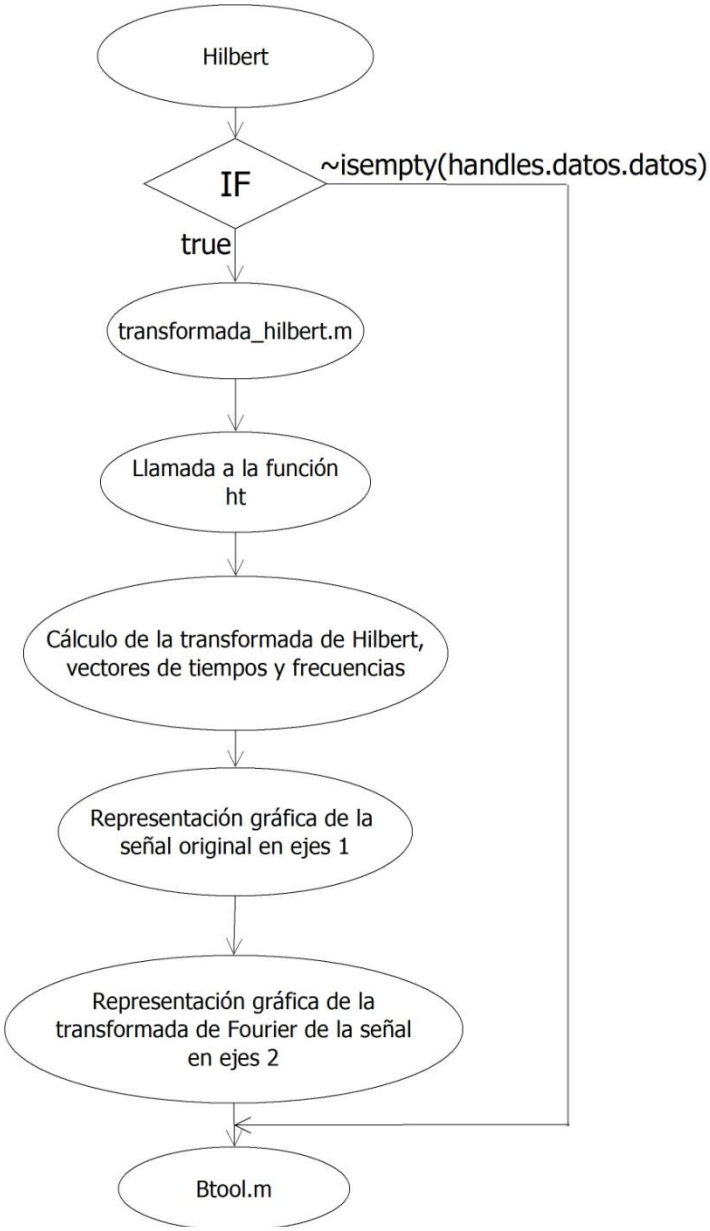


Figura 86. Diagrama de flujo de la función "transformada_hilbert.m"

Nivelesenergia.m

La función *"Nivelesenergia.m"* también tiene asociada un fichero de imagen "Nivelesenergia.fig", es decir, también se ejecuta una nueva interfaz gráfica al ejecutar dicha función. Sin embargo, al contrario de las otras dos funciones pertenecientes al menú de *"Transformadas"*, con esta interfaz gráfica si que el usuario puede interactuar con ella, teniendo opción a elegir entre las diferentes utilidades y funcionalidades que proporciona dicho panel de control. Véase *Figura 87*.

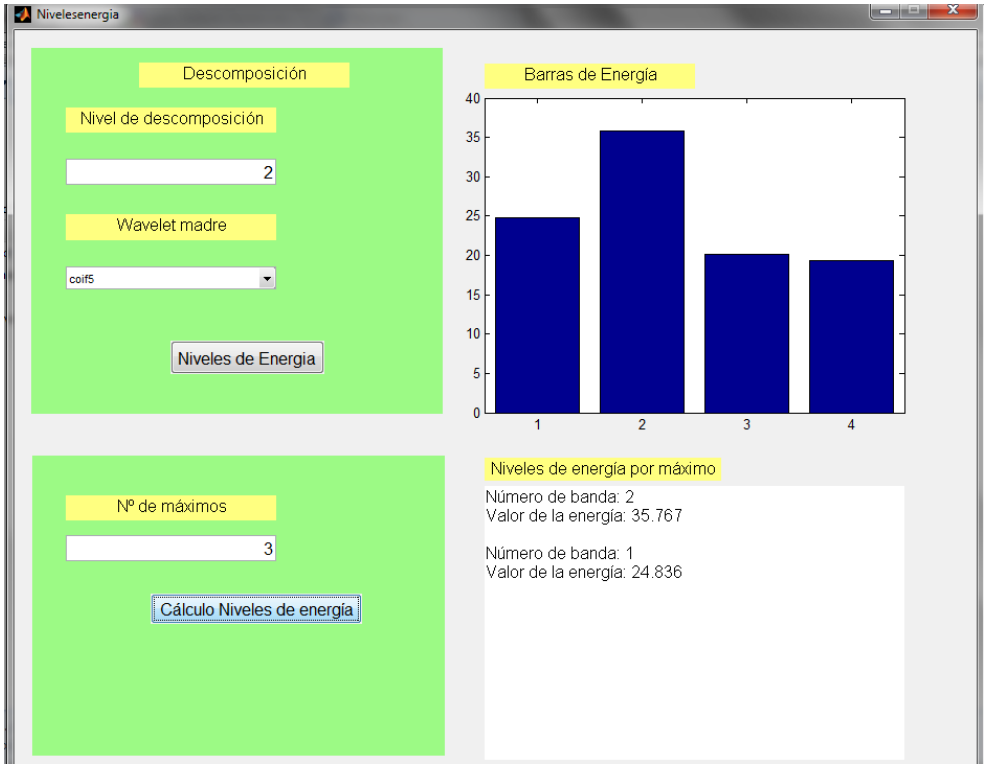


Figura 87. Interfaz gráfica de la función "Nivelesenergia.m"

El usuario puede elegir el número que desee de *"Nivel de descomposición"* y seleccionar una *"Wavelet madre"* entre la lista existente del menú desplegable. Una vez seleccionados dichos parámetros se pulsa el botón *"Niveles de Energía"* para realizar la representación del diagrama de barras de energía expuesto en la figura.

Del mismo modo, puede introducir el *"Nº de máximo"* que estime oportuno y pulsar el botón de *"Cálculo Niveles de energía"* para mostrar un en la figura un documento de texto explicativo de los parámetros introducidos y además representar gráficamente en otra figura nueva las diferentes formas de onda que adopta la señal adquirida. Véase *Figura 88*.

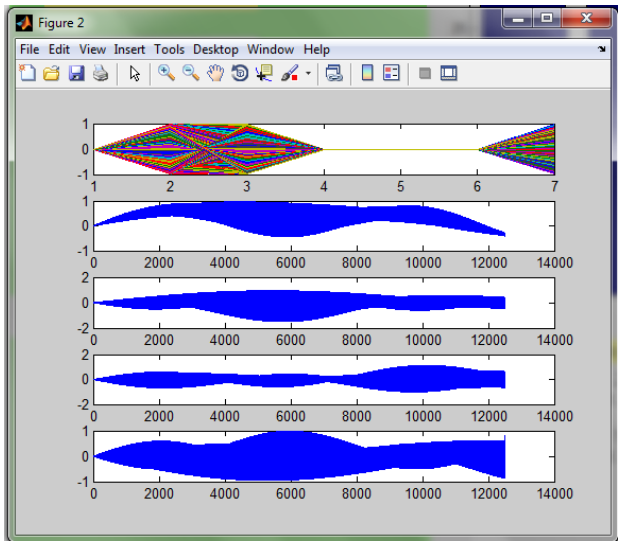


Figura 88. Representación de los diferentes niveles de energía

A continuación, se muestra el diagrama de flujo. Véase *Figura 89*.

Figura 89. Diagrama de flujo de la función "Nivelesenergia.m"

Por último, van a explicarse detalladamente las diferentes opciones que tiene el usuario en la interfaz gráfica principal *"Btool.m"* para la captación y adquisición de datos.

Los diferentes modos de trabajo que puede seleccionar el operario son el modo *"KEITHLEY"*, modo *"Micrófono"* o modo *"Simulación"*. Véase *Figura 90*.

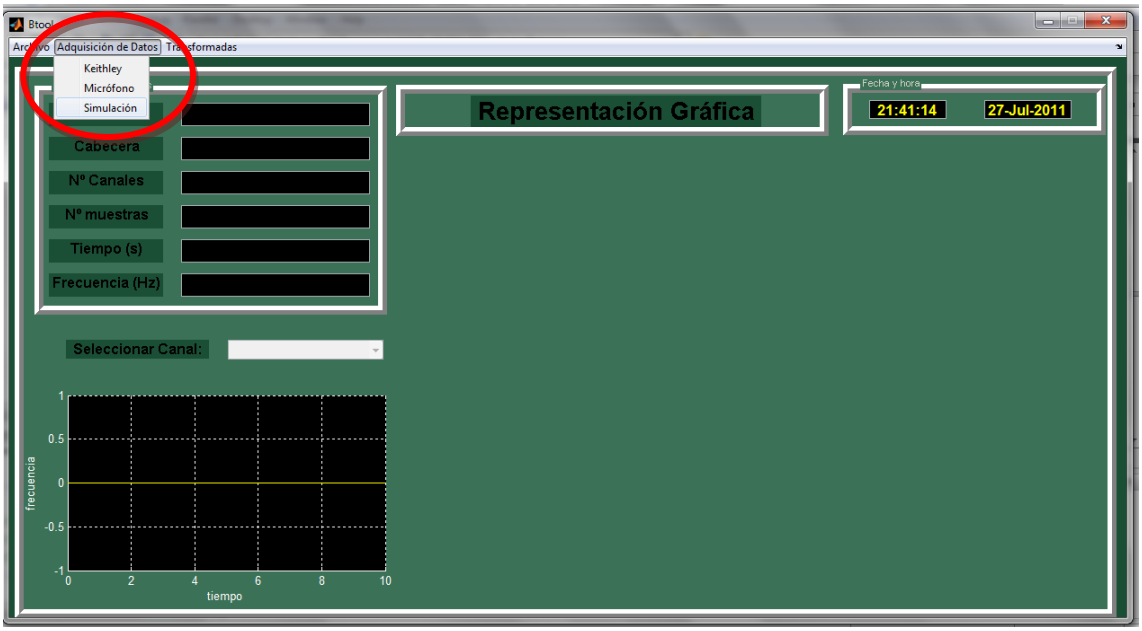


Figura 90. Menú de adquisición de datos de la interfaz gráfica de "Btool"

Cada uno de los modos de trabajo ejecuta en primer lugar una función propia para que determine los parámetros iniciales correctos para su perfecta ejecución. Posteriormente, una vez establecidos los condiciones necesarias todos los modos de trabajo desembocan en la ejecución de la misma función *"Allchannels.m"*.

Allchannels.m

Esta función es una interfaz gráfica ya que dispone de su correspondiente fichero de figura *"Allchannels.fig"*. Es la interfaz gráfica más importante de toda la herramienta porque es donde el usuario va realizar la mayoría de los trabajos e investigaciones. Esta interfaz dispone de multitud de opciones, funcionalidades y utilidades diversas para que el usuario pueda realizar una buena captación y adquisición de señal y acto seguido disponer de múltiples herramientas para poder realizar un buen tratamiento de la señal acorde a sus necesidades.

Hay que destacar que dependiendo del modo de trabajo que elija el usuario para la adquisición de señal, las opciones y funcionalidades correspondientes a dicho modo se activarán o desactivarán.

Sin embargo todas las señales para el tratamiento de señal serán comunes para los tres modos de trabajo y permanecerán activas siempre.

A continuación, se muestra la interfaz gráfica de *"Allchannels.m"* y se detallan cuales son las principales partes que el usuario necesita dominar para poder trabajar con la herramienta. Véase *Figura 91*.

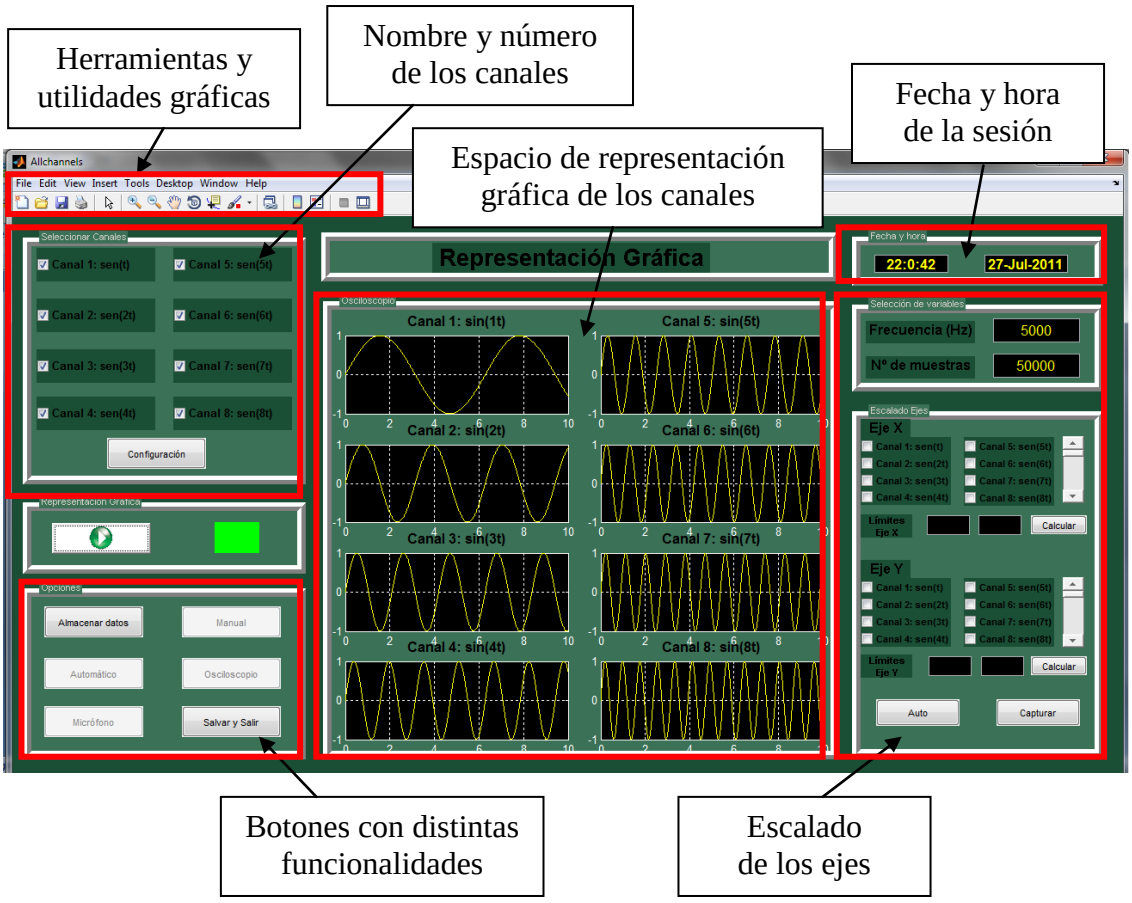


Figura 91. Panel de control de la interfaz gráfica de "Allchannels.m"

A continuación, se muestra el diagrama de flujo de la función *"Allchannels.m".* Véase *Figura 92*.

Figura 92. Diagrama de flujo de la función "Allchannels.m"

Captura_manual.m

La función *"Captura_manual.m"* tiene asociada un *".fig"*, por lo que dicha función dispone de una interfaz gráfica propia. Esta interfaz dispone de dos ejes destinados a lecturas y adquisición de datos. Véase *Figura 93*.

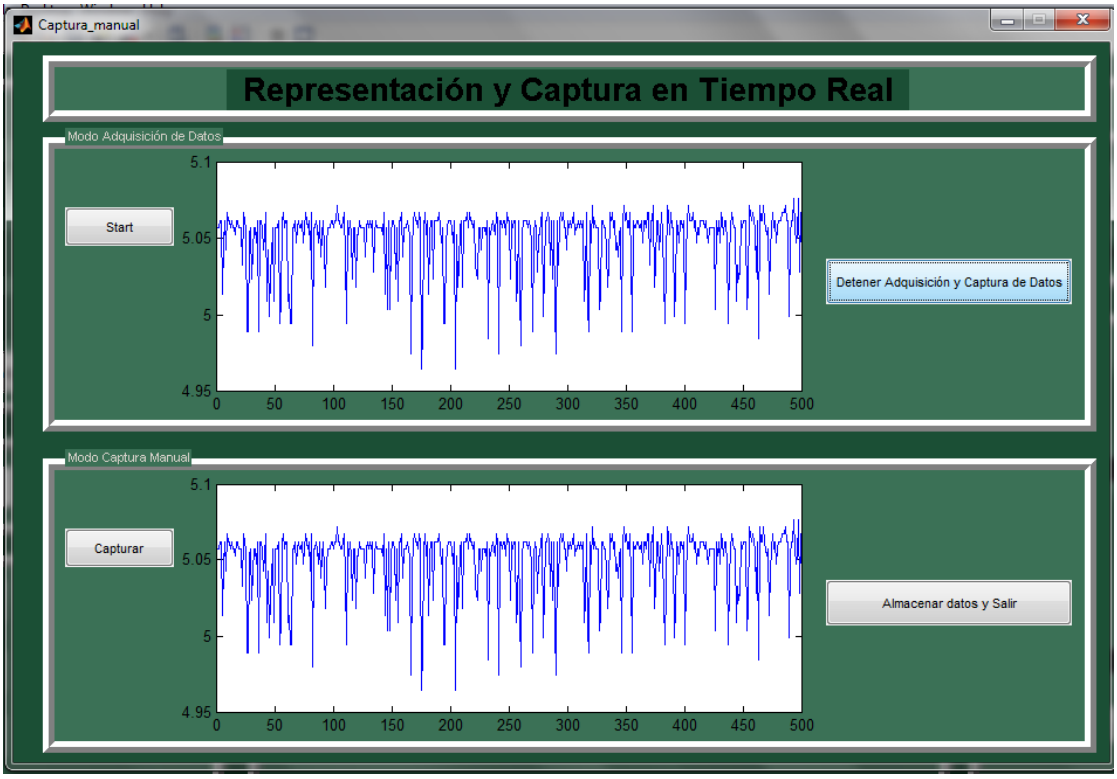


Figura 93. Interfaz gráfica de la función "Captura_manual.m"

En primer lugar, el usuario pulsa el botón *"Start"* y emerge una nueva interfaz gráfica diseñada para la calibración de los parámetros necesarios para la lectura y adquisición de datos.

A continuación, comienza la lectura de datos a través del dispositivo *"KEITHLEY"*. Esta lectura de datos se representa gráficamente en los ejes superiores. Una vez que el usuario determine que se ha producido el tiempo de estabilización de la señal de onda a captar, se pulsa el botón *"Capturar"* y a partir de ese justo momento comienzan a almacenarse los datos que el dispositivo está leyendo, representándolos gráficamente en los ejes inferiores.

Una vez adquiridos la cantidad de datos necesarios se pulsa el botón *"Detener Adquisición y Captura de Datos"* para finalizar la lectura de datos y congelar ambas representaciones gráficas hasta ese momento.

Por último, se pulsa el botón *"Almacenar datos y Salir"* para registrar los datos capturados y salir de la interfaz gráfica.

A continuación, se muestra el diagrama de flujo de la función *"Captura_manual"*. Véase *Figura 94*.

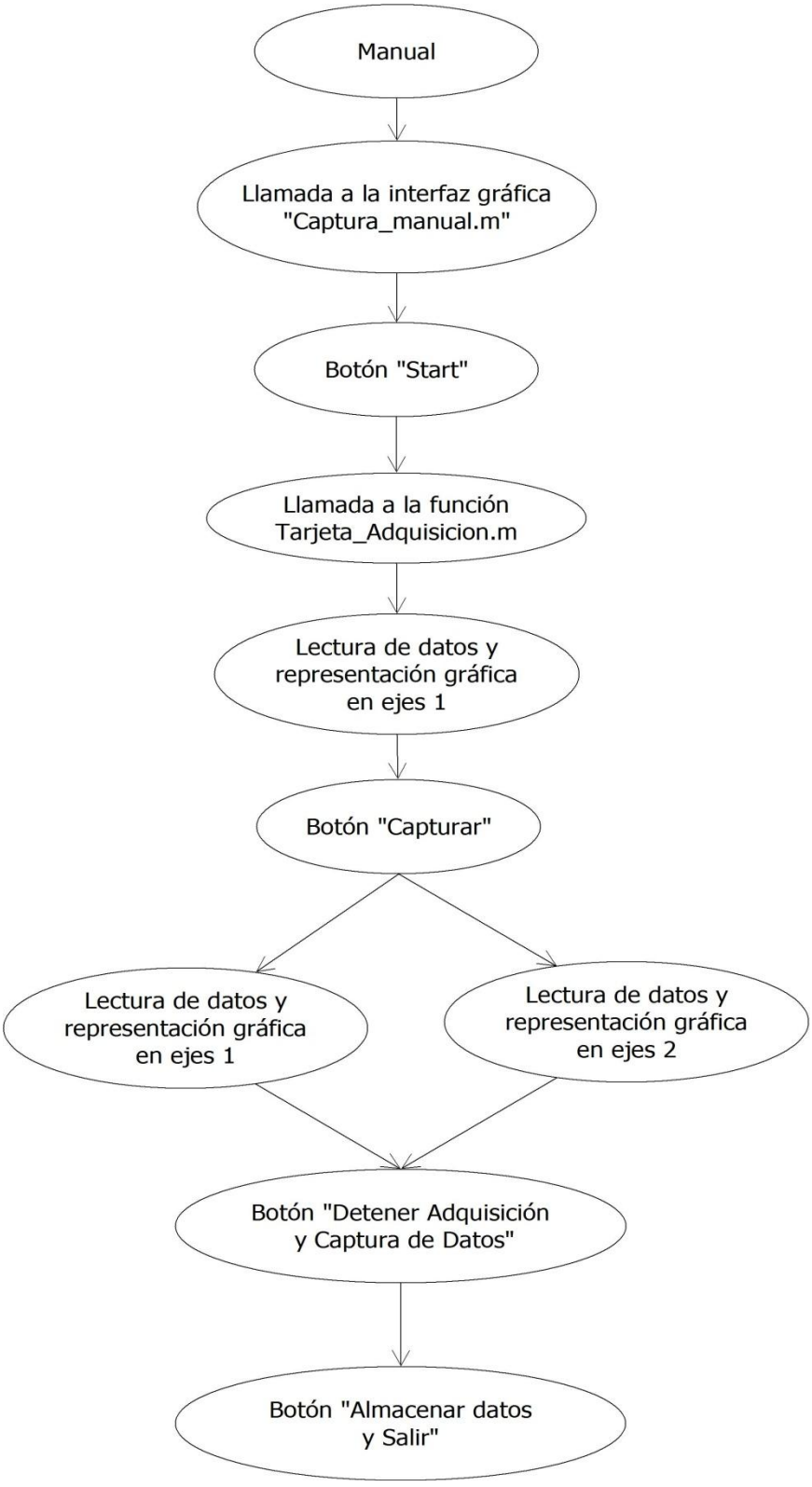


Figura 94. Diagrama de flujo de la función "Captura_manual.m"

Tarjeta_adquisicion.m

La función *"Tarjeta_adquisicion.m"* es una interfaz gráfica ya que dispone de su fichero de figura correspondiente *"Tarjeta_adquisicion.fig"*. Su única finalidad es que el usuario seleccione los canales donde se va a producir la adquisición de datos y acto seguido introduzca los parámetros correspondientes para poder realizar en las mejores condiciones posible la captura de la señal. Véase *Figura 95*.

Figura 95. Interfaz gráfica de la función "Tarjeta_adquisicion.m"

Configuracion.m

La función *"Configuracion.m"* es una interfaz gráfica ya que también dispone de su correspondiente fichero *".fig"*. Su objetivo es que el usuario pueda modificar el nombre de los canales según las necesidades para una mejor comprensión de las representaciones gráficas a realizar. Véase *Figura 96*.

Figura 96. Interfaz gráfica de la función "Configuracion.m"

4.3.2 Programación del software

La programación del software está estructurada por ficheros **.m* que están anidados entre sí y se van ejecutando unos u otros dependiendo de las instrucciones que vaya introduciendo el usuario por pantalla.

La descripción esquemática y el funcionamiento básico del sistema se han descrito anteriormente en el apartado 4.3.1.

A continuación, se explicará de forma detallada aspectos más internos de la programación del código para la compresión más sencilla de todas las funciones y comandos que puede llegar a realizar el software de la interfaz gráfica.

Declaración y asignación de variables globales

En primer lugar, cabe destacar la importancia de la declaración y asignación de variables globales a la hora de programar. Se declaran al principio de todos los ficheros **.m* donde se van a necesitar utilizarlas y su valor es visible en todos los espacios de trabajo de las funciones donde han sido declaradas.

Son muy útiles para realizar una programación rápida pero no se debe de abusar de ellas ya que disponen de un espacio de trabajo exclusivo y consumen muchos recursos. Esto provoca una pérdida de rendimiento provocando un funcionamiento ralentizado de la aplicación.

```
%%
%Declaración de variables globales
global hFig
global recuperar_microfono
global recuperar
global ai
%%
```

La asignación de variables globales se realiza de igual forma que las variables locales. Se establecen unas condiciones iniciales para dichas variables y el software las irá modificando según vaya sucediendo la ejecución del código.

Llamadas a funciones y funciones

En segundo lugar, hay que destacar tanto la importancia de las llamadas a las funciones como las propias funciones.

- **Llamadas a funciones**

  Las llamadas a funciones o funciones definidas se utilizan de la misma forma que las funciones propias de MATLAB, es decir, basta con llamar a la función con los parámetros necesarios desde la ventana de comandos o desde otra función o *"script"*.

Por ejemplo, la función definida en el Current Directory *"Allchannels.m"*, basta con realizar la llamada a la función con sus argumentos de entrada y salida desde cualquier otra función para que se ejecute su código.

```
%Llamada a la función "Allchannels" transmitiéndole como argumentos
de entrada las variables almacenadas en "prueba" y recibiendo como
argumentos de  salida las variables almacenadas en handles.datos
modificados por dicha función

[handles.datos]= Allchannels(prueba);
```

Valores de retorno

Argumentos de entrada

- **Funciones**

  La primera línea de un fichero que define una función tiene la forma:

  function [lista de valores de retorno] = nombre (lista de argumentos)

  donde *"nombre"* es el nombre de la función. Entre corchetes y separados por comas van las variables de salida o valores de retorno, y entre paréntesis también separados por comas las variables de entrada o argumentos.

Si no hay valores de retorno se omiten los corchetes y el signo igual. Si sólo hay un valor de retorno no hace falta poner corchetes. Tampoco hace falta poner paréntesis si no hay argumentos.

```
function [handles]=representacion(handles)

cont=0;
contador=0;
```

Variables locales "*cont*" y *"contador"*

Las variables definidas dentro de una función son variables locales, es decir, las variables que se crean dentro de la función pertenecen al espacio de trabajo de la función y son inaccesibles desde otras partes del programa. Además no interfieren con variables del mismo nombre definidas en otras funciones o partes del programa y desaparecen una vez finaliza la llamada a la función. Para que la función tenga acceso a variables que no han sido pasadas como argumentos es necesario declarar dichas variables como variables globales, tanto en el programa principal como en las distintas funciones que deben acceder a su valor.

Por lo tanto, una función tampoco puede acceder a las variables definidas en el espacio de trabajo o *"Workspace".* Para ejecutar una función es necesario que el directorio de trabajo coincida con el directorio donde se encuentra la función.

Una diferencia importante con la programación en C es que en MATLAB los argumentos de una función no se modifican nunca y los resultados se obtienen siempre a través de los valores de retorno, que pueden ser múltiples y matriciales. Tanto el número de argumentos como el de valores de retorno no son fijos, dependiendo de cómo se llama a la función. No hace falta calcular siempre todos los posibles valores de retorno de la función, sino sólo los que se espera obtener. Esto tiene consecuencias en términos de eficiencia y ahorro de tiempo de cálculo.

Ficheros de intercambio de datos

Este fichero tiene gran importancia ya que es el método de almacenamiento utilizado para guardar los valores de amplitud de señal en la función *"Osciloscipio".* Los comandos *"save"* y *"load"* utilizados en las funciones o *"scripts"* permiten guardar y cargar variables, matrices y vectores de forma selectiva en ficheros con un nombre especificado por el usuario.

Primeramente, se almacenan en el fichero **.mat* las variables, vectores o matrices utilizando el comando *"save"* y asignando un nombre a dicho fichero binario. Posteriormente, utilizando el comando *"load",* se carga del fichero binario **.mat* todos los valores de las variables, vectores y matrices y pueden ser utilizados para realizar otros cálculos u operaciones.

Este fichero binario tiene extensión **.mat* y se crea utilizando el comando:

save filename var1 var2 var3

donde "filename" es el nombre del archivo de intercambio de variables y "var1, var2, var3…" son los nombres de las variables que se desean guardar en dicho fichero.

Para recuperar las variables almacenadas en el archivo de intercambio de datos desde cualquier punto del programa se utiliza el comando:

load filename var1 var2 var3

donde "filename" es el nombre del archivo de intercambio de variables y "var1, var2, var3" es el nombre de las diferentes variables que se desean cargar del fichero.

Por tanto, una de las múltiples formas de comunicación de variables, vectores o matrices entre dos ficheros **.m* ya sean funciones o *"scripts",* se realiza mediante la creación de un fichero común con extensión **.mat*.

Posicionamiento de los ejes de representación gráfica

En primer lugar, hay que destacar que las coordenadas de posicionamiento para las diferentes representaciones gráficas sólo van a ser necesarias en dos interfaces gráficas, "*Btool.m*" y "*Allchannels.m*".

Cada una de estas interfaces va a tener un posicionamiento de los ejes de representación gráfica diferente ya que el propio diseño externo y el espacio destinado para la representación varía en cada una de ellas.

```
%-------------Llamada a la función posición_ejes.m------------------%

[handles.datos.pos1 handles.datos.pos2]=posicion_ejes(handles.datos);
%_____________________________________________________________________%
```

Tanto "*Btool.m*" como "*Allchannels.m*" se encargan de realizar la llamada a la función "*posicion_ejes.m*" cuya misión es introducir las coordenadas posicionales para referenciar cada representación gráfica en su lugar correspondiente.

```
%-----Posicionamiento de los ejes de representación gráfica --------%
% Definimos posiciones en interfaz para Allchannels
    pos1(1,:)=[0.30    0.645   0.43    0.14];
    pos1(2,:)=[0.30    0.45    0.43    0.14];
    pos1(3,:)=[0.30    0.255   0.43    0.14];
    pos1(4,:)=[0.30    0.06    0.43    0.14];

    pos2(1,:)=[0.30    0.645   0.2    0.14];
    pos2(2,:)=[0.30    0.45    0.2    0.14];
    pos2(3,:)=[0.30    0.255   0.2    0.14];
    pos2(4,:)=[0.30    0.06    0.2    0.14];
    pos2(5,:)=[0.53    0.645   0.2    0.14];
    pos2(6,:)=[0.53    0.45    0.2    0.14];
    pos2(7,:)=[0.53    0.255   0.2    0.14];
    pos2(8,:)=[0.53    0.06    0.2    0.14];

if datos.Btool==1  %Defimos las posiciones en interfaz para Btool

    pos1(1,:)=[0.37    0.68    0.61    0.15];
    pos1(2,:)=[0.37    0.47    0.61    0.15];
    pos1(3,:)=[0.37    0.26    0.61    0.15];
    pos1(4,:)=[0.37    0.05    0.61    0.15];

    pos2(1,:)=[0.37    0.68    0.29    0.15];
    pos2(2,:)=[0.37    0.47    0.29    0.15];
    pos2(3,:)=[0.37    0.26    0.29    0.15];
    pos2(4,:)=[0.37    0.05    0.29    0.15];
    pos2(5,:)=[0.69    0.68    0.29    0.15];
    pos2(6,:)=[0.69    0.47    0.29    0.15];
    pos2(7,:)=[0.69    0.26    0.29    0.15];
    pos2(8,:)=[0.69    0.05    0.29    0.15];
end
%_____________________________________________________________________%
```

Dependiendo del número de representaciones gráficas se utilizarán unas coordenadas u otras. Si el número de representaciones es inferior a cinco se aplicarán unas posiciones espaciales determinadas. Si por el contrario el número de representaciones es igual o superior a 5 se aplicarán otras totalmente diferentes.

```
%---------Representación gráfica según el número de elementos----------%
%Si la representación gráfica es menor de cinco elementos
if handles.datos.contador<5
handles.datos.ejes(i)=subplot('Position',handles.datos.pos1(contador,:));

else   %Si la representación gráfica es mayor o igual de cinco elementos
handles.datos.ejes(i)=subplot('Position',handles.datos.pos2(contador,:));

end
%_____________________________________________________________________%
```

El objetivo es que ambos posicionamientos se adapten completamente al espacio disponible en la interfaz gráfica para su visualización óptima.

El comando "*subplot*" es el encargado de crear los ejes y de seccionar el espacio destinado a la representación gráfica tomando los valores almacenados en la variable "handles.datos.ejes(i)". La variable "i" almacena el número de representaciones gráficas que el usuario a decidido realizar.

A continuación se muestra las diferentes posibilidades según se representen más o menos de cinco representaciones, tanto en "*Btool.m*" como en "*Allchannels.m*". Véase *Figura 97* y *Figura 98*.

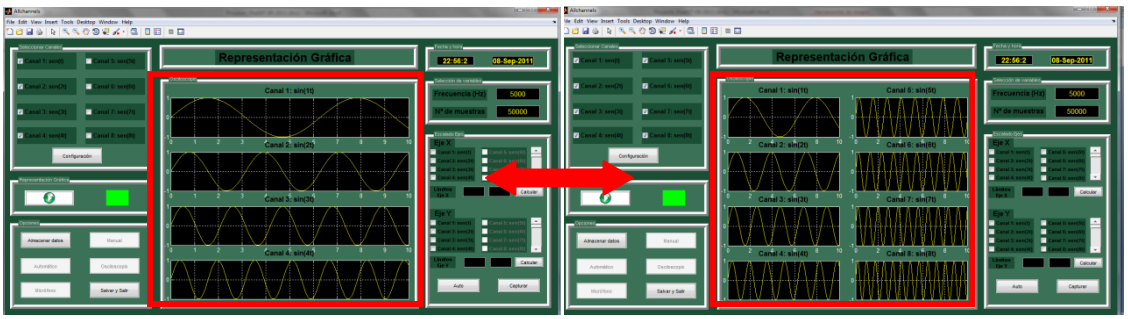


Figura 97. Representación en "Allchannels.m" según el número de gráficas

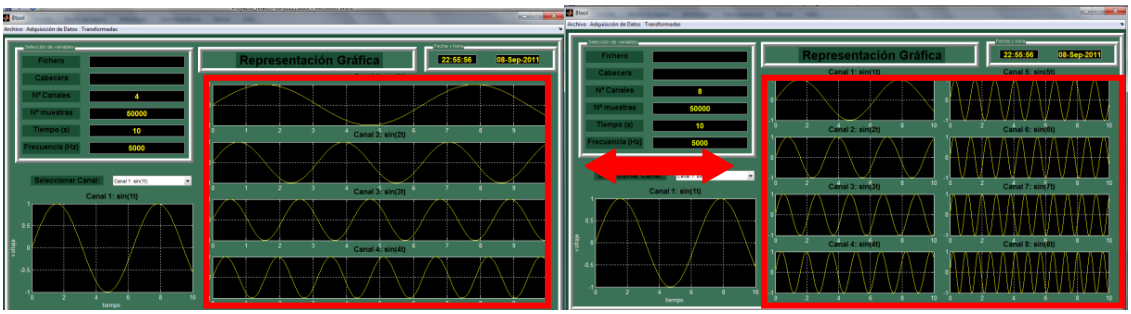


Figura 98. Representación en "Btool.m" según el número de gráficas

Programación de los "sliders" para el escalado de los ejes en "Allchannels.m"

Los "*sliders*" son las barritas que se encuentran situadas en la zona de escalado de los ejes que permiten aumentar o disminuir con precisión el tamaño de las gráficas representadas.

Éstos permiten realizar un escalamiento personalizado tanto del eje X como del eje Y de la gráfica. Sólo se permite realizar el escalamiento de un único canal con los "*sliders*" ya que cada gráfica puede disponer de diferentes parámetros y no se podrían obtener los resultados deseados escalando simultáneamente varios canales.

```
%----------Control de dimensionamiento de los sliders --------------%

for i=1:8
     if (handles.datos.canales_escalado_x(i))==1

set(handles.slider1,'Max',handles.datos.sliderX_sup(i)*handles.datos.
tmax)

handles.datos.limite_superiorX=handles.datos.sliderX_sup(i)*handles.d
atos.tmax;

set(handles.slider1,'Value',handles.datos.sliderX_sup(i)*handles.dato
s.tmax);

set(handles.slider1,'Min',handles.datos.sliderX_inf(i))

handles.datos.limite_inferiorX=handles.datos.sliderX_inf(i);

set(handles.slider1,'SliderStep',[0.02 0.2]);

     end
end
%_____________________________________________________________________%
```

Este bucle permite saber el canal de escalado seleccionado y calcular el rango de valores superior e inferior que tiene que adoptar el "*slider*" para poder realizar el escalado correcto con cada pulsación del "*slider*".

Una vez que se dispone del rango de valores adecuado, se diseña el algoritmo de cálculo para conseguir reducir o aumentar con cada pulsación del "*slider*" un incremento ó decremento determinado.

El algoritmo adoptado para cada pulsación es el siguiente:

```
%-----------Algoritmo de dimensionamiento de los sliders------------%

for i=1:8
        if get(eval(['handles.checkbox_x',num2str(i)]),'Value')==1

handles.datos.sliderX_sup(i)=(handles.datos.limite_medioX+((a-
handles.datos.limite_inferiorX)/2))/handles.datos.tmax;

handles.datos.sliderX_inf(i)=handles.datos.limite_inferiorX+((handles
.datos.limite_superiorX-a)/2);

set(handles.datos.ejes(i),'Xlim',[handles.datos.sliderX_inf(i)
handles.datos.sliderX_sup(i)*handles.datos.tmax],'Ylim',[handles.dato
s.sliderY_inf(i) handles.datos.sliderY_sup(i)]);

        end
end
%_____________________________________________________________________%
```

Programación de los "edit text" para el escalado de los ejes en "Allchannels.m"

Los "*edit text*" para realizar el escalado de los ejes tienen un funcionamiento mucho más sencillo que los "sliders". El manejo consiste en introducir exactamente el valor numérico superior e inferior del eje al que se quiere realizar el escalado.

Por tanto, ya no se necesita diseñar un algoritmo de incrementos y decrementos sino que ya se introduce directamente en la función de cálculo el valor introducido por el usuario y se realiza el escalado personalizado con dichos valores.

Además, se ha diseñado la función de cálculo de dicho escalamiento de los ejes para que no se produzcan errores, cubriendo todo tipo de posibilidades, ya que el usuario puede introducir multitud de caracteres que provoquen un error en el software al reconocer que no son valores numéricos adecuados. Un ejemplo sería que el usuario introdujera el límite superior más pequeño que el inferior ó que introdujera una letra en vez de un valor numérico.

A continuación se muestra la función diseñada para el control de errores que se pueden producir al introducir valores en los "*edit text*":

```
%---------------Control de error de los "edit text"-------------------%
%Obtengo el valor del edit text para el escalado del eje X
limite_superiorX=str2double(get(handles.edit5,'String'));
limite_inferiorX=str2double(get(handles.edit8,'String'));

%Evaluo la entrada del edit text y si no es un número muestra un aviso
if isnan(limite_superiorX)==1
    uiwait(warndlg('Introduzca un valor numérico para el límite del eje
X superior','Advertencia'));
    set(handles.edit5,'String','');
    return
end

%Evaluo la entrada del edit text y si no es un número muestra un aviso
if isnan(limite_inferiorX)==1
    uiwait(warndlg('Introduzca un valor numérico para el límite del eje
X inferior','Advertencia'));
    set(handles.edit8,'String','');
    return
end

%Evaluo entrada del edit text y si es número negativo muestra un aviso
if (limite_inferiorX)<0
    uiwait(warndlg('Introduzca un valor mayor o igual que 0 para el
límite del eje X inferior','Advertencia'));
    set(handles.edit8,'String','');
    return
end

%Evaluo entrada del edit text y si es número negativo muestra un aviso
if (limite_superiorX)<=0
    uiwait(warndlg('Introduzca un valor mayor que 0 para el límite del
eje X superior','Advertencia'));
    set(handles.edit5,'String','');
    return
end

%Evaluo la entrada del edit text y si limite sup<inf muestra un aviso
if limite_inferiorX>=limite_superiorX
    uiwait(warndlg('No puede ser el límite superior del eje X menor que
el límite inferior del eje X','Advertencia'));
    set(handles.edit5,'String','');
    return
end

%_____________________________________________________________________%
```

A continuación se muestra la función diseñada para el escalamiento de los ejes de las gráficas mediante los "*edit text*":

```
%----------Función de dimensionamiento de los "edit text"-----------%

for i=1:8

handles.datos.canales_escalado_x(i)=get(eval(['handles.checkbox_x',nu
m2str(i)]),'Value');

if (handles.datos.canales_escalado_x(i))==1
handles.datos.sliderX_sup(i)=limite_superiorX;

handles.datos.sliderX_sup(i)=handles.datos.sliderX_sup(i)/handles.dat
os.tmax;

handles.datos.sliderX_inf(i)=limite_inferiorX;

set(handles.datos.ejes(i),'Xlim',[handles.datos.sliderX_inf(i)
handles.datos.sliderX_sup(i)*handles.datos.tmax],'Ylim',[handles.dato
s.sliderY_inf(i) handles.datos.sliderY_sup(i)]);
end

end

%_____________________________________________________________________%
```

Funcionamiento del botón "Capturar" en "Allchannels.m"

Este botón trata básicamente de almacenar los valores de los ejes de las gráficas después del escalado, ya sea mediante los "*sliders*", "*edit text*" o por la función de "*Zoom*" de la barra de herramientas para poder posteriormente realizar la representación de las gráficas en la interfaz "*Btool.m*" y mantener los valores del escalado realizado por el usuario.

Si no se pulsa el botón *"Capturar"* después de un escalamiento no se almacenan los límites de los ejes introducidos y se perderá el trabajo y el tratamiento de las señales que se hayan realizado.

Este bucle obtiene los canales de escalado seleccionados y almacena los valores de las posiciones de escalado de los ejes X e Y para posteriormente poder representarlas gráficamente en *"Btool.m"* con su correspondiente dimensionamiento de cada gráfica.

```
%--------------Programación del botón "Capturar"--------------------%
for i=1:8

handles.datos.canales_escalado_x(i)=get(eval(['handles.checkbox_x',nu
m2str(i)]),'Value');

handles.datos.canales_escalado_y(i)=get(eval(['handles.checkbox_y',nu
m2str(i)]),'Value');

    if (handles.datos.canales_escalado_x(i) ||
handles.datos.canales_escalado_y(i))==1
        contador=contador+1;
        c=get(handles.datos.ejes(i),'XLim');
        d=get(handles.datos.ejes(i),'YLim');
        handles.datos.sliderX_inf(i)=c(1);
        handles.datos.sliderX_sup(i)=c(2);

handles.datos.sliderX_sup(i)=handles.datos.sliderX_sup(i)/handles.dat
os.tmax;
        handles.datos.sliderY_inf(i)=d(1);
        handles.datos.sliderY_sup(i)=d(2);
    end
end

%_____________________________________________________________________%
```

Programación del algoritmo de adquisición de datos en "Captura_manual.m"

La función de "*Captura_manual.m*" permite realizar una lectura continua de datos y empezar a almacenar datos simultáneamente cuando el usuario estime oportuno. Para conseguir este efecto es necesario que la tarjeta de adquisición de datos consiga realizar dos funciones a la vez, leer y registrar en tiempo real. Esto es físicamente imposible ya que dicha tarjeta no está preparada para ello.

Sin embargo, se utiliza una función interna que alterna el leer y el registrar cada un intervalo de tiempo determinado por el usuario. Primero hay que realizar la configuración correcta de la tarjeta de adquisición "KEITHLEY":

```
%------------Configuración tarjeta de adquisición de datos----------%
set(ai0,...
   'TriggerRepeat', 1,...
   'TriggerType', 'Manual',...
   'SampleRate',handles.datos.sample_rate,...
   'SamplesPerTrigger',handles.datos.samples_trigger,...
   'BufferingConfig', [10000 10],...
   'TimerPeriod',0.1,...
   'Timerfcn', @localTimerAction);
%_____________________________________________________________________%
```

El parámetro "*Timerfcn, @localTimerAction*" es el que permite ejecutar su función relacionada cada un intervalo de tiempo determinado, en este caso, cada 0,1 segundos.

Sin embargo, una vez que esté configurada la tarjeta de adquisición para que se ejecute automáticamente dicha función, hay que diseñar un algoritmo dentro de dicha función que permita realizar la representación en tiempo real de los datos capturados a partir de que el usuario ejecute la orden de captura.

El algoritmo trata en primer lugar de representar en la primera gráfica en tiempo real los datos que se están leyendo. Una vez que el usuario estime oportuno pulsar el botón de *"Capturar"* se representarán en las dos gráficas la misma información. Sin embargo, se empezarán a almacenar en la variable *"datos_parciales"* tanto los datos como los tiempos desde ese mismo instante para poder posteriormente realizar un registro de dicho datos y descartar la información irrelevante del proceso de adquisición de datos.

```
%---------Algoritmo que permite la representación de datos ---------%

function data = localTimerAction(ai0,event)

if contador==1
    contador=0;
    tic; %Instrucción que permite calcular tiempos
end

x= peekdata(ai0,500); %Permite capturar datos en tiempo real
h=findobj('Name', 'Captura_manual');
hejes=findobj(h,'Type', 'axes');

if ~isempty(x)
    if detener1==0
        subplot(hejes(2));
        plot(x); %Representación en la primera gráfica
    end
    if detener2==0
        if contador==0
            contador=2;
            tiempo_parcial=clock; %Calcula el tiempo transcurrido
        end
        y=x;
        subplot(hejes(1));
        plot(y); %Representación en la segunda gráfica
        datos_parciales=[datos_parciales,y'];%Almacenamiento de datos
    end                                       parciales

    datos_totales=[datos_totales,x'];%Almacenamiento de datos totales
end

%_____________________________________________________________________%
```

Funcionamiento del botón "Micrófono" en "Allchannels.m"

El botón "Micrófono" permite realizar una adquisición de datos durante un tiempo especificado por el usuario a través de la tarjeta de sonido integrada del ordenador.

Esta tarjeta de sonido no sólo sirve para capturar datos provenientes exclusivamente de un micrófono analógico sino que cualquier dispositivo adecuado que se conecte a la clavija correspondiente del micrófono puede transmitir su información registrada. Un ejemplo sería un acelerómetro.

Para empezar, hay que configurar la tarjeta de sonido adecuadamente para poder realizar la captura del sonido proveniente del micrófono u otro dispositivo.

```
%------------------Configuración tarjeta de sonido------------------%

%Se comprueba que la tarjeta del micrófono se encuentra activa
     ai = analoginput('winsound');
     %Se añaden dos canales
     addchannel(ai, [1 2]);
     %Se ejecuta la función "demoai_ftt"
     demoai_fft

     % Configure the object to acquire 2 seconds of data at 8000 Hz.
     Fs = handles.datos.frecuencia;
     duration = handles.datos.timer_period;
     set(ai, 'SampleRate', Fs);
     set(ai, 'SamplesPerTrigger', duration*Fs);
     %%
%_____________________________________________________________________%
```

Acto seguido, una vez realizada la configuración se activa el modo de adquisición y se ejecuta el comando "*getdata(ai)*" que permite el almacenamiento de los datos.

```
%---------------Programación del botón micrófono--------------------%
        % Start the acquisition and retrieve the data.
        start(ai);
        handles.datos.canales=[1 1 0 0 0 0 0 0];

        %Comienza la captura de datos durante el tiempo determinado
        [handles.datos.datos_micro,handles.datos.t] = getdata(ai);
        handles.datos.datos_micro=handles.datos.datos_micro';
        handles.datos.lectura_real=handles.datos.datos_micro;
        handles.datos.t=handles.datos.t';

        %Borra y limpia los datos almacenados en la variable "ai"
        delete(ai);   clear ai;
%_____________________________________________________________________%
```

Programación del fichero "cargar.m"

En primer lugar se consigue el nombre y la ruta del fichero a cargar para obtener la ruta completa. Acto seguido, se realiza la apertura del fichero para tener acceso a su contenido y se comprueba si dispone de cabecera o no.

Si dispone de cabecera se comienza a leer el fichero que dispone de una estructura determinada previamente y se asocia cada valor a su variable correspondiente. Después se realiza la lectura de los canales seleccionados y por último se almacenan los datos estructurados por columnas, donde cada columna se asocia con un canal y se termina la lectura del fichero.

```
%-------------------Cargar un archivo o fichero----------------------%
%Obtengo la ruta del fichero a cargar
[nombre,ruta] = uigetfile({'*.dat','All Points-Files
(*.dat)'},'CARGAR REGISTRO');
if nombre==0
    return
else
    handles.datos.boton_graficar=1;
    fichero = fullfile(ruta,nombre);
    handles.datos.nombre_fichero=nombre;
    %Obtenemos el identificador del fichero con la funcion fopen,
    %especificando la ruta del fichero y en formato escritura
    handles.datos.fid=fopen(fichero,'r');
end

%Con fgets apuntando al identificador guardamos en "a" la siguiente
línea del archivo que leemos
a=fgets(handles.datos.fid);
if a(1)=='*'
    handles.datos.cabeceras='Si';

    %Si el archivo tiene un asterisco,abre el archivo lee su primera
linea y calcula el numero de columnas.
    if strcmp(handles.datos.cabeceras,'Si')==1
        [l1,l2]=size(a);
        b=str2num(a(2:l2));
        handles.datos.contador=b(1);
        handles.datos.muestras=b(2);
        handles.datos.frecuencia=b(3);
        handles.datos.canales=zeros(1,8);
        handles.datos.titulos_fichero=[];

        for i=1:handles.datos.contador
            a=fgets(handles.datos.fid);
            [l1,l2]=size(a);
            canales(i)=str2num(a(9));
            etiqueta=strcat(a(12:l2));
            handles.datos.canales(canales(i))=1;
            %Guardo los titulos del fichero sin ordenar
            handles.datos.titulos_fichero{i}=sprintf('Canal %s :
            %s',num2str(canales(i)),etiqueta);
        end
```

```
        %Ordeno los titulos del fichero
        contador=0;
        for i=1:8
            if handles.datos.canales(i)==1
                contador=contador+1;
                handles.datos.titulos_fichero_ordenado{i}=
                cell2mat(handles.datos.titulos_fichero(contador));
            else
                handles.datos.titulos_fichero_ordenado{i}=0;
            end
        end

        %Almaceno en el vector el nombre de los canales del fichero
        handles.datos.lectura_datos=fscanf(handles.datos.fid,'%f'
,[handles.datos.contador,handles.datos.muestras]);
        %Cierro el fichero de texto
        fclose(handles.datos.fid);
%_____________________________________________________________________%
```

A continuación, se muestra una imagen con la estructura de almacenamiento de las diferentes variables y datos en los ficheros de texto. Este formato en común tanto para la labor de registro de información como de lectura de la misma. Véase *Figura 99*.

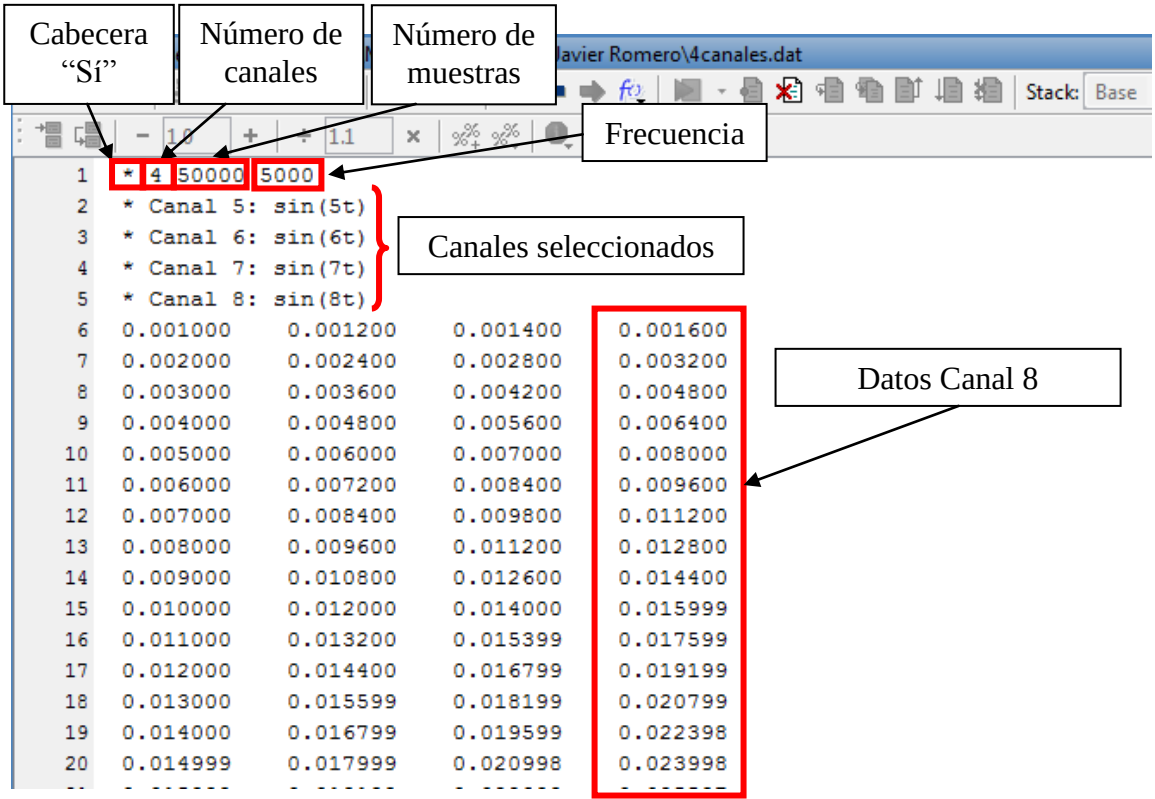


Figura 99. Estructura de almacenamiento y lectura de los ficheros de texto

Si por el contrario, el fichero no dispone de cabecera, se muestran directamente las columnas con los datos almacenados para realizar su lectura.

Programación del fichero "registros.m"

Antes de empezar se especifica la ruta donde se quiere almacenar el fichero de registros. Una vez dispongamos de la ruta, abrimos el fichero para empezar a registrar la información.

Se almacena el número de canales, número de muestras y frecuencia. Acto seguido, se guarda el nombre de todos los canales seleccionados por el usuario. Finalmente se almacenan por columnas todos los datos registrados.

```
%---------------Programación del botón micrófono--------------------%
%Creo un fichero.dat donde almaceno el canal seleccionado
[nombre,ruta]=uiputfile({'*.dat','All Points-Files(*.dat)'},'GUARDAR
REGISTRO');
    if nombre==0
        return
    else
        fichero = fullfile(ruta,nombre);
%Obtenemos el identificador del fichero con la funcion fopen,
%especificando la ruta del fichero y en formato escritura
        fid=fopen(fichero,'w');
%Calculamos el número de filas y columnas de los datos a almacenar
fprintf(fid,'%s %s %s %s\r\n','*',num2str(handles.datos.contador),
num2str(handles.datos.muestras),num2str(handles.datos.frecuencia));
contador=0;
     for i=1:8
         if handles.datos.canales(i)==1,
         fprintf(fid,'%s %s\r\n','*',char(handles.datos.titulos(i)));
         end
      end
%Almacena en cadena tantos (%f /t) como número de canales menos 1
% /t tabulador horizontal
        cadena='';
        if handles.datos.contador>1
            for i=1:handles.datos.contador-1
                cadena=strcat(cadena,'%f \t');
            end
        end
%Almacena en cadena tanto (%f \t) como canales y el ultimo termino
%concatena (%f\r\n)
        cadena=strcat(cadena,'%f\r\n');
        contador=0;
%Ordeno los datos a representar en filas seguidas
        for i=1:8
            if handles.datos.canales(i)==1
                contador=contador+1;
                datos(contador,:)=handles.datos.datos(i,:);
            end
        end
        fprintf(fid,cadena,datos);
        fclose(fid);
    end
%_____________________________________________________________________%
```

5. Resultados obtenidos

5.1 Arquitectura hardware

5.1.1 Sistema de ensayos: Machine Fault Simulator Lite (MFS Lite)

Máquina de ensayo de rodamientos. La máquina ha sido diseñada por la empresa Spectra Quest, Inc. para el ensayo de rodamientos y elementos rotatorios y puede ser utilizada a altas velocidades por lo que en todo momento el funcionamiento se hará con la pantalla de seguridad bajada. Véase *Figura 100*.

El equipo consta de:

- Motor eléctrico trifásico Marathon "four in one" modelo DV3.
- Variador de frecuencia.
- Tacómetro.
- Rodamientos en los que efectuar el estudio.
- Juego de cargas de distintos pesos y dimensiones.
- Juego de acoples flexibles o rígidos.

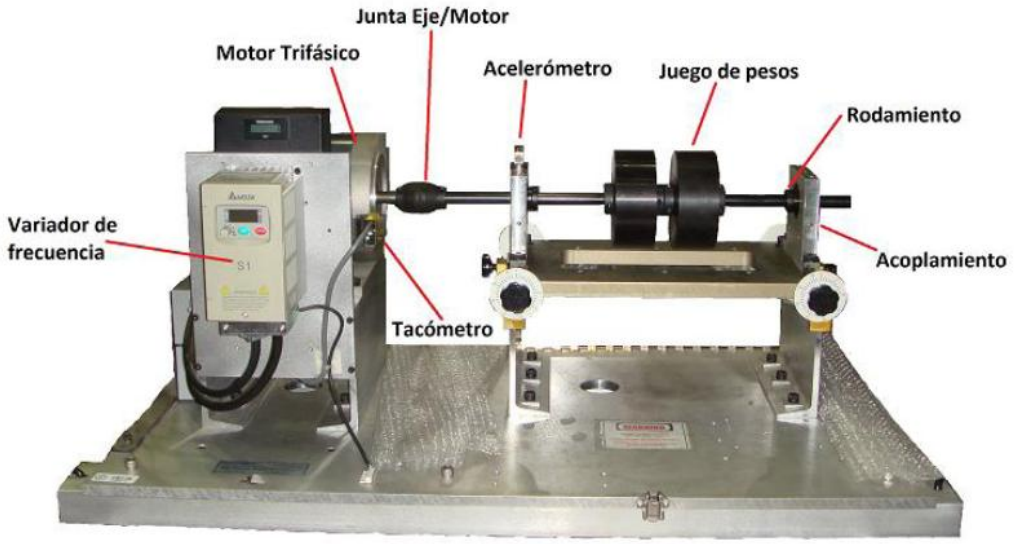


Figura 100. MFS Lite

El *"MFS Lite"* permite realizar los ensayos a distintas velocidades de rotación controladas de forma manual o bien, programar diferentes tipos de rampas controladas de manera automática. Esto último permite simular el comportamiento del elemento en estudio, bien sea un eje o bien un rodamiento, ante un arranque como el que tendría lugar en cualquier máquina a la que estuviese destinado el elemento en cuestión.

En lo que respecta al diseño del *"MFS Lite"*, cabe decir que se trata de una máquina de ensayos muy sencilla en cuanto a construcción se refiere, tal y como puede apreciarse en la *Figura 100*. Dicha máquina consta de una bancada sobre la que se apoyan diferentes elementos fijos, como puede ser el motor, y los elementos objeto de estudio como pueden ser ejes o rodamientos.

5.1.2 Acelerómetro

El acelerómetro es empleado para medir la aceleración absoluta en las vibraciones de un elemento. Consiste en una lámina de material piezoeléctrico que, al estar sometido a compresión mecánica o tensiones de corte, genera cargas eléctricas en las caras, proporcionales a la fuerza aplicada.

El modelo KS943B.10 de la empresa MMF es un acelerómetro triaxial de pequeñas dimensiones adecuado para objetos ligeros con un taladro central que permite fijarlo al elemento sobre el que se colocará dentro del sistema y su correcta alineación de los ejes. Véase *Figura 101*.

Puede experimentar tres tipos de deformación mecánica:

- Compresión.
- Flexión.
- Cizallamiento o deformación de corte.

Para el caso del que se ocupa el presente proyecto se ha empleado un acelerómetro triaxial de la marca MMF modelo KS-943B.10. Véase *Figura 101*.

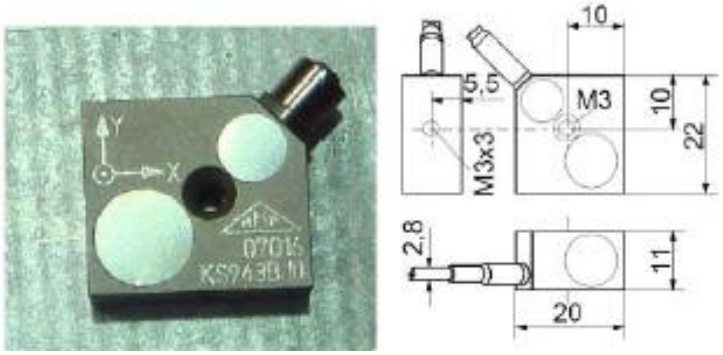


Figura 101. Acelerómetro triaxial modelo KS-943B.10

El acelerómetro empleado ocupará tres canales de la aplicación Btool que podrán ser representados en tiempo real y son:

- Eje x.
- Eje y.
- Eje z.

Con lo que se ofrece la posibilidad de medir las vibraciones mecánicas que se den en el sistema a lo largo de tres direcciones, contribuyendo así a obtener unos resultados que den una idea mucho más completa del estado del sistema.

5.1.3 Tarjeta de adquisición de datos

La tarjeta de adquisición de datos es de la marca Keithley modelo KUSB 3100. Su función es la de convertir la señal analógica adquirida por los sensores en una señal digital para su posterior procesado en un ordenador. Véase *Figura 102*.

Figura 102. Tarjeta de adquisición de datos

La conexión plug-and-play permite que el PC detecte automáticamente el módulo de adquisición de datos KUSB-3100 cuando es conectado al ordenador y busque el software necesario para su funcionamiento.

Las tarjetas de la serie KUSB-3100 son externas al ordenador. Esta localización externa proporciona beneficios en el rendimiento para dispositivos sensibles al ruido tales como sistemas de adquisición. Sin embargo, pueden ser susceptibles de picos de baja, ESD, sobrecargas y otras condiciones dañinas. Estos picos pueden causar fallos en el sistema, incluso daños permanentes al ordenador.

Las características principales de la tarjeta de adquisición de datos se muestran en la siguiente figura. Véase *Figura 103*.

Descripción	Bajo coste, multifunción
Resolución	12 bit
Tasa de transferencia	50 kS/s
Canales de entrada analógica	8 SE
Canales de salida analógica	2
Canales digitales I/O	16
Contador/Reloj	1
Ganancia	1, 2, 4, 8

Figura 103. Características de la tarjeta de adquisición de datos Keithley KUSB

5.1.4 Filtros y amplificadores

El propósito de un filtro es remover señales indeseadas desde la señal que se está trabajando. Con el propósito de poder usar las señales tomadas con el acelerómetro, es necesario conectar éste último a unos filtros acondicionadores de señal que preparan la señal para su posterior captura mediante una tarjeta de adquisición de datos. En este proyecto, se han empleado cuatro módulos acondicionadores MMF M32, véase *Figura 104*, los cuáles son unos acondicionadores de señal para transductores con salida compatible IEPE. Estos módulos proporcionan la fuente de alimentación necesaria para el circuito electrónico del sensor. La unidad ofrece tres rangos de ganancia y un filtro paso bajo conectable.

Un filtro para ruido es empleado en el estudio de señales continuas, tales como por ejemplo la temperatura, ya que atenúan señales de alta frecuencia debido a que estas podrían reducir la precisión de las mediciones, repercutiendo en un empeoramiento de los resultados del estudio que se esté llevando a cabo.

Las señales alternas, tales como vibración a menudo requieren un tipo diferente de filtros conocidos como un filtro anti-aliasing. Igual que un filtro de ruido, el filtro anti-aliasing es también un filtro pasa bajos. Sin embargo, se debe tener una tasa de truncado muy abrupta, es decir que remueva completamente todas las frecuencias de la señal que son más altas que la entrada de ancho de banda de la tarjeta. Constan de un potenciómetro que permite amplificar la señal. Véase *Figura 104*.

Figura 104. Filtro-amplificador

Los módulos acondicionadores IEPE M32 contienen el circuito electrónico para el suministro de un sensor. Para aplicaciones multicanal estos módulos se pueden conectar entre sí mediante banana plugs enroscadas en un lateral del módulo. Estas conexiones conectan la fuente de alimentación a todos los módulos.

5.1.5 Tacómetro

Junto con un elemento reflectante dispuesto en el eje del rotor, permite registrar la velocidad de giro del motor. A continuación se muestra un tacómetro digital. Véase *Figura 105*.

Figura 105. Tacómetro digital

5.1.6 Computador

El equipo de medida irá conectado, a través de la tarjeta de adquisición de datos y mediante un conector USB a un PC, en el cual estará cargada la aplicación *"Btool"* diseñada para éste fin.

Para saber más acerca del software empleado, consultar *Capítulo 4*, en el que se detalla extensamente la aplicación *"Btool"* junto con todas las posibilidades que ofrece para el estudio de las vibraciones mecánicas entre otras posibles perturbaciones de los ejes y rodamientos.

5.2 Descripción de los ensayos de trabajo

En este capítulo se procederá a describir las características de los ensayos realizados y comentar todos los parámetros considerados para el correcto desarrollo de los estudios efectuados. También se detallarán la utilización de la herramienta *"Btool"* empleada para la adquisición, monitorización y tratamiento de las señales adquiridas.

5.2.1 Descripción de los ensayos con "MFS Lite"

Los ensayos realizados consisten en una prueba de arranque tal y como tendría lugar en cualquier situación real y que es un caso crítico ante la presencia de una fisura en un eje. Esta prueba consiste en generar una rampa, previamente programada en el controlador del banco de ensayos, con una duración de 10 segundos, de los cuáles los ocho primeros corresponde a la etapa de aceleración, hasta llegar a 80 Hz (4800 rpm), y a partir de ahí, los siguientes segundos pertenecen a la etapa de velocidad constante, tal y como se muestra en el gráfico siguiente. Véase *Figura 106.*

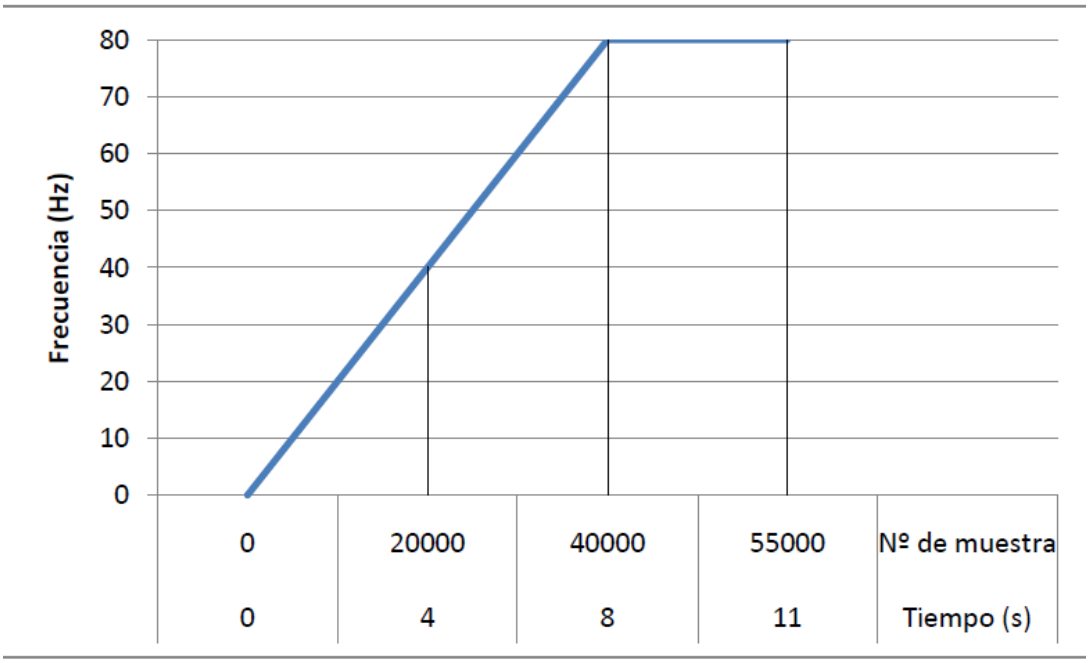


Figura 106. Rampa de ensayo

La toma de adquisición de datos tiene que iniciarse previamente al lanzamiento del *"MFS Lite"* ya que es necesario este margen para salvar el desfase existente y necesario entre el inicio de la toma de medidas por parte de este programa y el inicio de la ejecución del ensayo por parte del banco de ensayos, puesto que ambos hay que iniciarlos de forma manual y no se ha realizado la

automatización del mismo en este proceso. Por tanto, se puede programar la frecuencia de muestreo y el número de muestras por segundo de la tarjeta de adquisición de datos desde la aplicación *"Btool"* para adaptarse y muestrear perfectamente la frecuencia de trabajo del *"MFS Lite"* durante cada ensayo realizado.

Además, hay que añadir el hecho de que antes de tomar cualquier medida válida, se ha procedido previamente a mantener, durante un cierto tiempo, la máquina funcionando a una velocidad aproximada de 20 Hz (1200rpm) con el fin de que todos los elementos del *"MFS Lite"* se encuentren en condiciones óptimas de funcionamiento.

También hay que considerar que los filtros empleados se han fijado con una ganancia unidad para la velocidad y de diez para las medidas de aceleración correspondientes a los ejes X, Y y Z. La orientación de estos tres ejes X, Y y Z según el sistema cartesiano viene definida por la colocación del acelerómetro en el banco de ensayos. La disposición del acelerómetro y el sistema de coordenadas puede verse desde una vista frontal en la siguiente figura. Véase *Figura 107*.

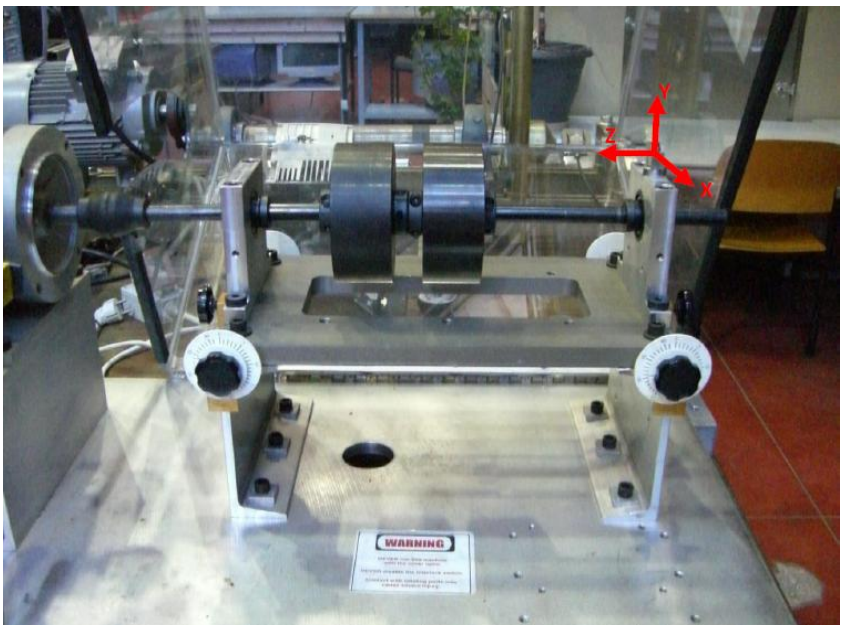


Figura 107. Orientación de los ejes en el sistema

Se ha analizado el comportamiento de tres ejes de las mismas características salvo por la presencia de fisuras generadas en algunos de ellos. Estos ejes están fabricados con aluminio con un diámetro de 16 mm y una longitud de 500 mm, con una distancia entre apoyos de 350 mm. Las fisuras mecanizadas en los ejes son fisuras transversales en la zona central de los mismos. Véase *Figura 108*.

En total, se han analizado tres ejes como los que aparecen en la figura y que se corresponden con un:

- Eje sin fisura.
- Eje con una fisura correspondiente al 12.5 % de su diámetro.
- Eje con una fisura del 25 % de su diámetro.

Figura 108. Ejes con diferentes tanto por ciento de fisuras

Antes de definir los ensayos a realizar es necesario calcular la velocidad crítica del eje sano para evitar trabajar cerca de ella o superarla, ya que este hecho podría provocar la rotura del eje o, en su defecto, daños en el mismo que lo inutilizarían para seguir cumpliendo su función dentro del conjunto. Esto se debe a la deformación sufrida durante la rotación, la cuál es una función de la velocidad y que presenta sus máximos cuando la velocidad de giro del eje coincide con alguna de sus velocidades críticas.

De estas velocidades críticas, la primera de ellas es la que se considera más importante, aunque en algunos casos la segunda también tiene su transcendencia, debido a que son las que se pueden encontrar dentro de las velocidades de operación de la máquina, mientras que las otras son tan elevadas que no se podría llegar a alcanzarlas. Teniendo esto en cuenta, para la configuración empleada en este proyecto, la velocidad crítica teórica es de 8026 rpm (134 Hz).

Además, se puede observar que esta velocidad crítica teórica del eje dista bastante de la velocidad alcanzada durante los ensayos (4800 rpm (80 Hz)), con lo que se evita la influencia de los efectos de dicha velocidad crítica en los resultados, incluso ante la presencia de fisuras en los ejes.

5.2.2 Descripción de los ensayos con generador de funciones

Los ensayos consisten en realizar una serie de pruebas utilizando todos los modos de funcionamiento de la interfaz gráfica *"Btool",* capturando y monitorizando todo tipo de ondas ya sean sinusoidales, cuadráticas, de rampa, pulsos ó continua. Además, se utiliza un amplio rango de frecuencias y de amplitudes de onda, pudiendo ir variándose dinámicamente en tiempo real.

Los ensayos mediante el generador de funciones han sido un objetivo prioritario para poder diseñar y depurar la programación del software de la interfaz gráfica y así poder adaptarla más fácilmente a un entorno de trabajo real.

Los elementos necesarios para la realización del ensaño es uno o varios generadores de señales, la tarjeta de adquisición *"KEITHLEY"* y un ordenador donde se carga el software de la interfaz gráfica. Véase *Figura 109*.

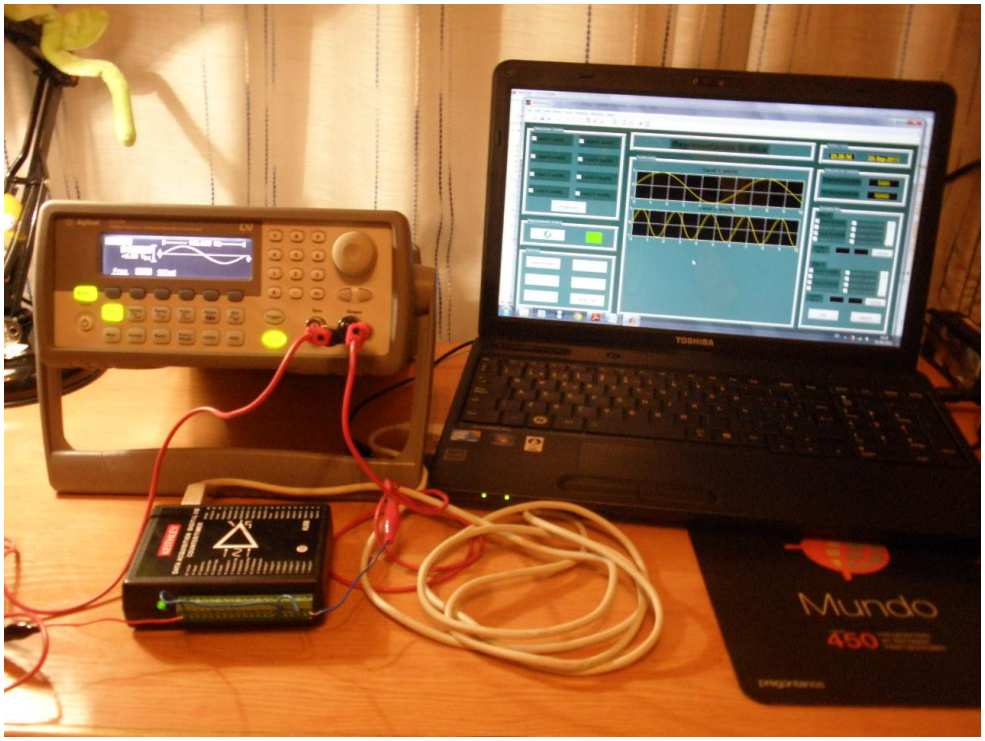


Figura 109. Componentes para el ensayo con el generador de funciones

Hay que destacar que de cada generador de funciones sólo se puede obtener una única señal. Por tanto, se necesitan varios generadores de señal y fuentes de continua para poder introducir diferentes señales en la tarjeta de adquisición de datos. La propia tarjeta *"KEITHLEY"* dispone de una salida propia de +5V de tensión continua y otra de 2,5V de tensión de referencia por lo que también se pueden utilizar como señales de entrada.

A continuación, se muestra una figura donde se puede observar el conexionado que se lleva a cabo para una toma de datos, apreciándose cómo se conecta cada una de las fuentes a las entradas analógicas de la tarjeta de adquisición de datos. Véase *Figura 110.*

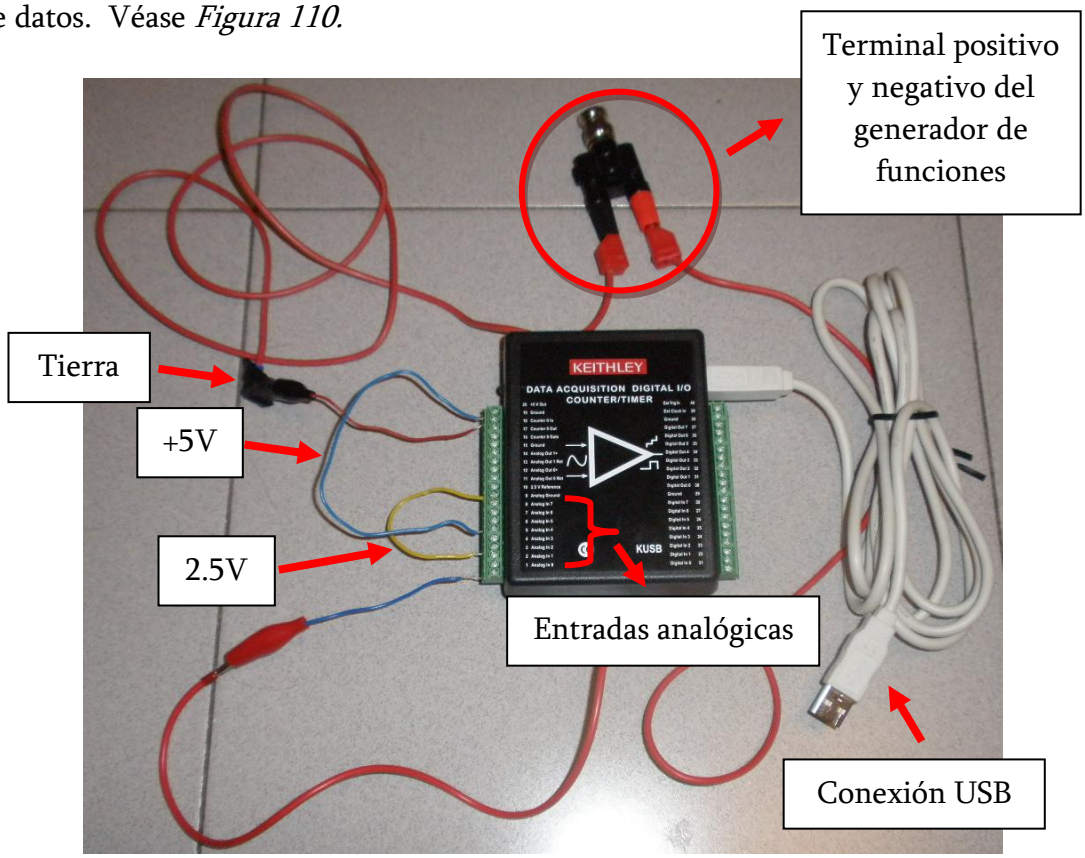


Figura 110. Conexionado de la tarjeta de adquisición de datos

A cada fuente le corresponde una entrada analógica de la tarjeta de adquisición de datos, y posteriormente, se han de puentear todas las tierras de cada generador y conectarlas con la tierra.

Por tanto, una vez conectados todos los generadores de funciones y fuentes de continua a la tarjeta de adquisición de datos *"KEITHLEY"*, y ésta con su conector USB al ordenador, tendremos concluido el montaje previo a la realización de los ensayos de corrección y depuración de código software. Véase *Figura 111.*

En la *Figura 111* se conectan mediante generadores de señal alterna y continua, cinco de los ochos canales disponibles de los que dispone la tarjeta de adquisición "KEITHLEY" que se descomponen de la siguiente manera:

- Dos fuentes de DC a diferentes niveles de tensión.
- Dos generadores de funciones funcionando en régimen de AC con distintas amplitudes y frecuencias.
- Un generador de funciones como fuente de una señal cuadrada.

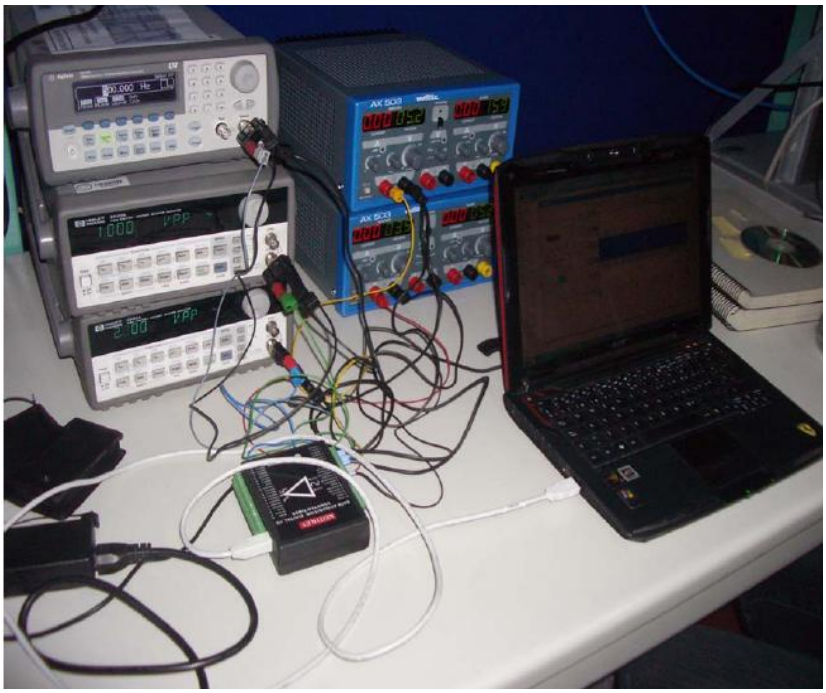


Figura 111. Montaje previo a la realización de los ensayos

Para realizar el ensayo mediante los generadores de funciones y las fuentes de continua, la interfaz gráfica *"Btool"* dispone de tres modos de funcionamiento diferentes.

En primer lugar, mediante el *"Modo simulación"* el usuario no puede capturar las ondas introducidas por el generador de funciones, sólo puede utilizar las ondas sinusoidales creados por defecto en el propio software como entrenamiento.

En segundo lugar, mediante el *"Modo micrófono"* el usuario puede capturar datos a través de la tarjeta de sonido integrada en el ordenador, conectando cualquier tipo de dispositivo que se adapte a la clavija del micrófono. Por tanto, tampoco puede capturar las ondas provenientes del generador de funciones.

Por último, el *"Modo KEITHLEY"* es el indicado para la captura, almacenamiento y representación de las señales analógicas o digitales del generador de funciones entrantes por la respectiva tarjeta de adquisición de datos *"KEITHLEY".*

Dentro del modo de funcionamiento "KEITHLEY" el usuario dispondrá de varias opciones de captación de datos diferentes para poder elegir dependiendo de la finalidad del objetivo para el que se realiza la captura de datos. Las opciones disponibles son:

- Modo *"Automático".*
- Modo *"Manual".*
- Modo *"Osciloscopio".*

5.3 Manual de uso

5.3.1 Inicialización de la interfaz gráfica externa

Para iniciar el programa que ejecuta la interfaz gráfica externa se puede inicializar desde el propio programa MATLAB o se puede generar un ejecutable para lanzar la interfaz gráfica sin la necesidad de tener instalado el programa completo.

En primer lugar, para iniciar la interfaz gráfica desde MATLAB hay que seguir los pasos detallados a continuación.

1. Ejecutar MATLAB.

2. Abrir en el "Current Directory" la carpeta específica de trabajo del software de la interfaz gráfica.

3. Escribir en el "Command Window" el nombre del fichero que inicializa el sistema sin la extensión **.m* y pulsar *ENTER*.

En segundo lugar, para crear una aplicación ejecutable (Standalone application) hay que seguir una serie de pasos.

1. Ejecutar MATLAB.

2. Abrir en el "Current Directory" la carpeta específica de trabajo del software de la interfaz gráfica.

3. Escribir en el "Command Window": *mcc -m "Btool.m"* y pulsar *ENTER* para generar la aplicación ejecutable (Standalone application) en el directorio actual de la carpeta específica de trabajo dentro del "Current Directory".

4. Para ejecutar la aplicación que se acaba de generar hay dos métodos posibles.

   En primer lugar, ejecutar directamente la aplicación ejecutable desde la propia carpeta de trabajo donde se ha generado pulsando sobre ella doble clic ó pulsando botón secundario y eligiendo la opción "Open".

   En segundo lugar, ejecutar el "Command Prompt" y especificar la ruta completa donde se encuentra la aplicación ejecutable y escribir el nombre del archivo con la terminación **.exe*.

A continuación, se muestra una imagen de la interfaz gráfica principal "*Btool.m*" donde se ha realizado una representación de ocho canales, con una longitud de onda diferente para su posterior evaluación y análisis. Véase *Figura 112*.

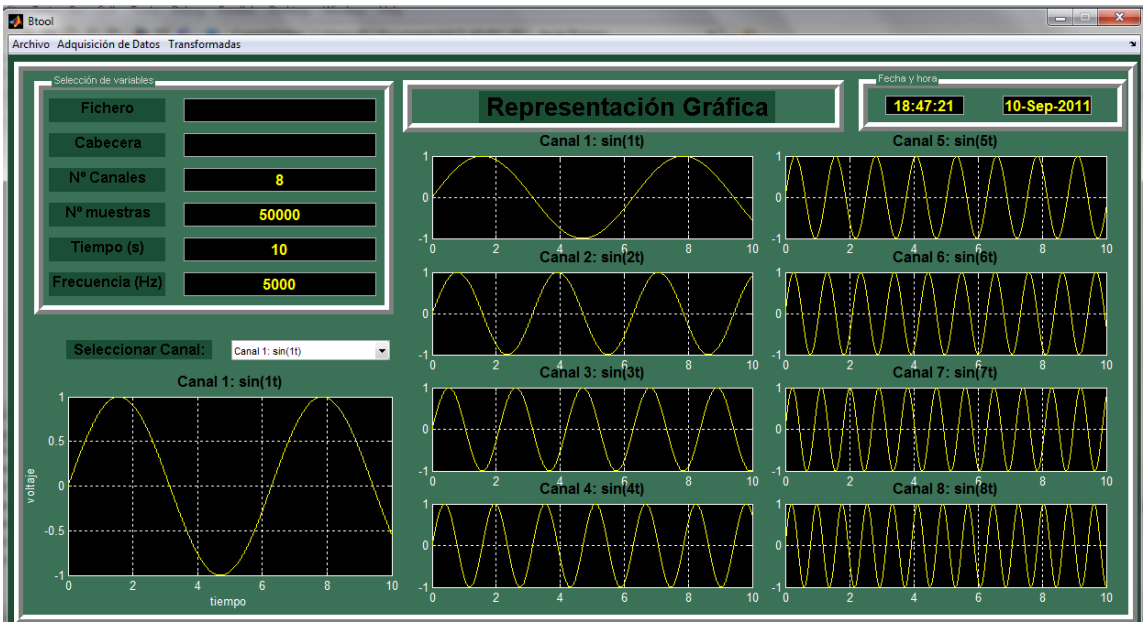


Figura 112. Interfaz gráfica "Btool.m"

A continuación, se muestra una imagen de la interfaz gráfica de "*Allchannels.m*" donde se ha realizado una captura y representación de cuatro canales. Además esta interfaz permite la posibilidad de utilizar diferentes herramientas para realizar un tratamiento de cada una de las señales según las conveniencias del usuario. Véase *Figura 113*.

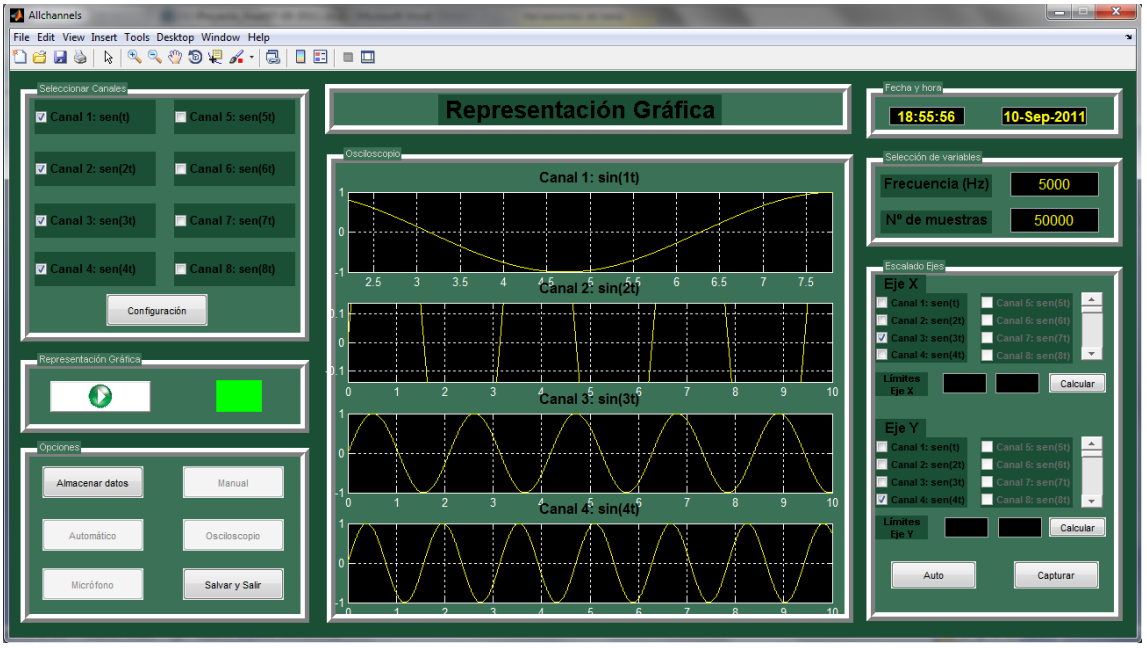


Figura 113. Interfaz gráfica "Allchannels.m"

5.3.2 Funcionamiento de la interfaz gráfica externa

Conceptos previos antes de iniciar la ejecución de la interfaz gráfica

1. Antes de ejecutar el programa principal, se deben realizar todas las conexiones físicas necesarias para poder llevar a cabo los requerimientos de la sesión de trabajo. Conexión de la tarjeta *"KEITHLEY"* al puerto USB, conexión de las diversas señales de onda entrantes a los respectivos terminales de dicha tarjeta *"KEITHLEY"* y conexión a la clavija del micrófono el dispositivo de adquisición de datos seleccionado para la sesión de trabajo.

2. Una vez que se haya inicializado la función de *"Btool.m"* y se ejecute la interfaz gráfica externa, lo primero que se debe hacer es realizar una captura de datos utilizando el menú contextual de "Adquisición de Datos" para elegir uno entre los posibles métodos disponibles ó realizar una carga de un fichero de una sesión de trabajo ya almacenada anteriormente mediante el menú contextual *"Archivo"*. No se puede realizar un almacenamiento de fichero o realizar un tratamiento de señal mediante el menú contextual *"Transformadas"* porque todavía la interfaz gráfica no ha realizado la adquisición de ninguna señal y por tanto no dispone de señales o datos almacenados.

3. Para realizar escalamientos de las señales de onda representadas en la interfaz gráfica "*Allchannels.m*" se debe tener un único canal seleccionado para utilizar los "*sliders*" y se pueden tener varios canales seleccionados para utilizar los "*edit text*". Sin embargo, en éstos últimos, no se pueden introducir valores incoherentes, caracteres no numéricos o valores que no estén comprendidos en un rango determinado. Del mismo modo, tanto el botón "*Auto*" como el botón "*Capturar*" sólo tendrán algún efecto en aquellos canales que se encuentren seleccionados.

4. La interfaz gráfica "*Tarjeta_adquisicion.m*" se utiliza tanto para la configuración de la tarjeta de adquisición *"KEIHTLEY"* como para la configuración de la tarjeta de sonido del ordenador. Los parámetros de configuración no se pueden elegir arbitrariamente sin conocimiento previo ya que se producirían errores. Por este motivo, se dispone del botón "*Auto*" que carga una serie de valores predeterminados para realizar un funcionamiento correcto de la aplicación. Además, para el caso de la configuración de la tarjeta *"KEITHLEY"* se necesita añadir manualmente los canales por los que se desea recibir datos ya que ésta no los detecta automáticamente.

5. Para realizar la lectura y representación de las señales entrantes a la tarjeta *"KEITHLEY"* en la interfaz gráfica del osciloscopio digital se pulsa el botón "*Trigger*". No se debe cerrar la interfaz del osciloscopio digital sin previamente antes haber pulsado el botón "*Stop*" para detener la lectura de las señales.

6. En la interfaz gráfica "*Captura_manual.m*" la secuencia lógica de funcionamiento es:
    1. Pulsar el botón "*Start*" para realizar la lectura de las señales entrantes a la tarjeta *"KEITHLEY".*
    2. Pulsar el botón "*Capturar*" para realizar el almacenamiento de la señal de entrada a partir de dicho momento.
    3. Pulsar el botón "*Detener Adquisición y Captura de Datos*" para detener tanto la lectura como el almacenamiento de la señal.
    4. Pulsar el botón "*Almacenar datos y Salir*" para guardar toda la información recogida y abandonar dicha interfaz gráfica y volver a "*Allchannels.m*".

    El resto de posibilidades de actuación diferentes a dicha secuencia lógico provocarían un error en el sistemas que obligaría a reiniciar de nuevo la aplicación y perder la sesión de trabajo actual.

Descripción todos los valores paramétricos que monitoriza la interfaz gráfica

- **Fichero:** nombre de un archivo de datos almacenado en otra sesión de trabajo.
- **Cabecera:** distinción que tienen los archivos de datos para saber si poseen ala comienzo una estructura con variables almacenadas o no.
- **Nº de canales:** número de representaciones gráficas realizadas de los canales.
- **Nº de muestras:** número de muestras que se almacenan para cada representación gráfica.
- **Tiempo [s]:** duración de la adquisición de datos.
- **Frecuencia [Hz]:** magnitud que mide el número de repeticiones por unidad de tiempo de las señales representadas.
- **Fecha y hora:** indica la fecha y hora actualizada en cada sesión de trabajo.

Funcionamiento y aplicación de los botones de la interfaz gráfica

El botón *"Configuración"* de la interfaz gráfica de *"Allchannels.m"* permite a los usuarios cambiar el nombre de los canales que vayan a utilizar para poder diferencia claramente cual son la variables que se están representando en cada gráfica. Además, los nombres de las gráficas modificados también quedan almacenados en los ficheros de texto cuando se guarda una sesión de trabajo.

El botón *"Almacenar datos"* de la interfaz gráfica de *"Allchannels.m"* se pulsa una vez que se han realizado las representaciones gráficas y el usuario ha concluido de utilizar las herramientas de modificación de las señales. Este botón permite almacenar en un fichero de texto toda la información de cada una de las representaciones gráficas que se han dibujado en la interfaz con sus variables más significativas.

El botón "Salvar y Salir" de la interfaz gráfica de *"Allchannels.m"* se pulsa cuando el usuario ha terminado de realizar todas las acciones correspondientes en la interfaz gráficas y desea enviar su sesión de trabajo a la interfaz gráfica *"Btool.m"* donde dispone de nuevas funcionalidades.

El botón *"Trigger"* de la interfaz gráfica del osciloscopio digital se pulsa cuando el usuario desea que éste empiece a realizar la lectura y representación gráfica de la señal entrante a la tarjeta de adquisición de datos *"KEITHLEY".* Una vez que se desea parar dicha representación se pulsa el botón *"Stop"* situado el mismo botón. Por tanto, el mismo botón dependiendo de la función que se esté realizando en dicho momento tendrá una finalidad distinta.

Del mismo modo, si el sistema se encuentra en pleno proceso de monitorización de parámetros y gráficas y se pulsa el botón *"Salir",* el software cierra la aplicación y no guarda los datos almacenados hasta ese momento.

Funcionamiento y aplicación de los componentes de la paleta de herramientas

En la *Figura 114*, se muestran todos los componentes de la paleta de herramientas integrada en la interfaz gráfica externa.

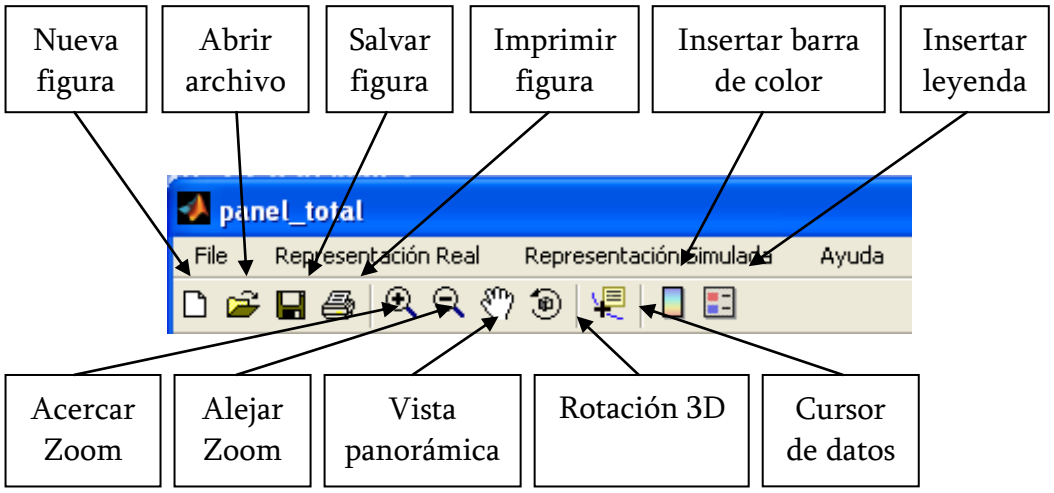


Figura 114. Paleta de herramientas de la interfaz gráfica externa

A continuación, se va describir la funcionalidad de los iconos más útiles para la adquisición de datos y tratamiento de señal:

1. **Acercar/Alejar Zoom**

Si se pulsa el botón de *"Acercar Zoom"* o *"Alejar Zoom"* sobre los ejes de representación gráfica permite realizar un acercamiento o alejamiento automático donde se pulse con el cursor.

Si se pulsa el segundo botón del ratón sobre los ejes se despliega un menú secundario que permite manejar algunas opciones interesantes. Véase *Figura 115*. A continuación se muestran dichas opciones:

- **Zoom in/out:** tiene la misma funcionalidad que al realizar un clic con el botón principal en la gráfica. Permite un acercamiento o alejamiento automático según donde se pulse con el cursor.

- **Reset to Original View:** permite regresar a los ejes a su posición inicial de visión establecida.

- **Unconstrained Zoom:** permite realizar un zoom de acercamiento o alejamiento a una región o área seleccionada X-Y con el cursor.

- **Horizontal Zoom (2-D Plots Only):** permite realizar un zoom de acercamiento o alejamiento a una sección X con el cursor.

- **Vertical Zoom (2-D Plots Only):** permite realizar un zoom de acercamiento o alejamiento a una sección Y con el cursor.

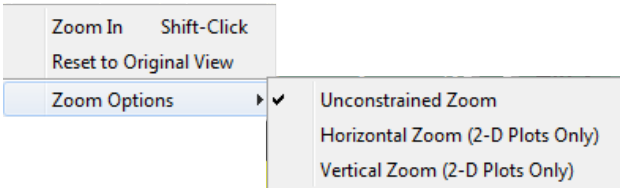


Figura 115. Menú contextual de la herramienta de "Zoom in/out"

2. **Cursor de datos**

Si se pulsa el botón de "Cursor de datos" se sustituye el puntero del ratón por una cruceta que permite crear una etiqueta con coordenadas (X, Y, Z) de la posición seleccionada con tan sólo pulsar con el botón izquierdo del ratón sobre cualquier posición de la flecha simbólica representada.

Si se pulsa el segundo botón sobre los ejes se despliega un menú secundario que permite manejar diversas opciones. Véase *Figura 116* y *Figura 117*.

- **Create New Datatip:** crea una nueva etiqueta de coordenadas.

- **Delete Currente Datatip:** elimina la etiqueta de coordenadas seleccionada.

- **Delete All Datatips:** elimina todas las etiquetas de coordenadas.

- **Export Cursor Data to Workspace:** al seleccionar una etiqueta se pueden exportar sus coordenadas al Workspace.

- **Edit Text Update Function:** edita una función para actualizar las coordenadas.

- **Select Text Update Function:** selecciona una función para actualizar las coordenadas.

En el menú secundario "Selection Style", es decir, selección de estilo, se despliega un sub-menú con las siguientes opciones:

- **Mouse Position:** selecciona la posición marcada por el puntero del ratón para indicar las coordenadas espaciales.
- **Snap to Nearest Data Vortex:** selecciona la posición marcada más cercana a un vértice para indicar las coordenadas espaciales.

En el menú secundario "Display Style", es decir, estilo del display, se despliega un sub-menú con las siguientes opciones:

- **Window Inside Figure:** la etiqueta con las coordenadas espaciales se crea dentro de los ejes de representación.
- **Datatip:** la etiqueta con las coordenadas espaciales se crea en una nueva ventana en el exterior de la interfaz gráfica.

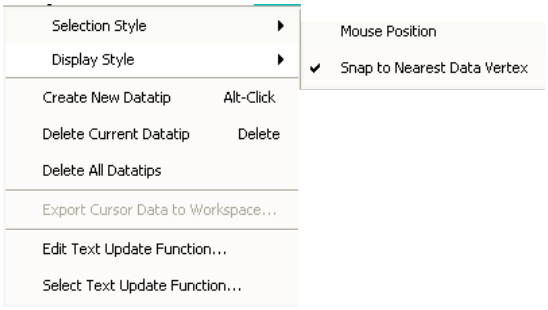


Figura 116. Menú contextual de la herramienta "Cursor de datos" (1)

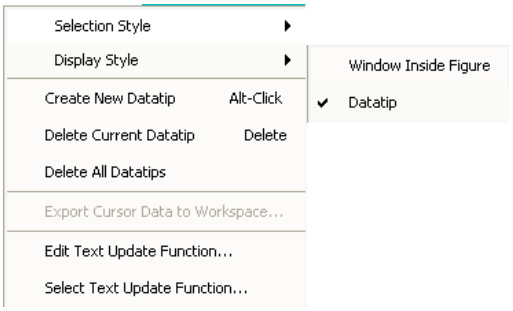


Figura 117. Menú contextual de la herramienta "Cursor de datos" (2)

3. Vista panorámica

Si se pulsa el botón de "Vista panorámica" se sustituye el puntero del ratón por una mano que permite cambiar el punto de vista horizontal y vertical de los ejes con tan sólo mantener pulsado el botón izquierdo del ratón y mover el puntero por cualquier punto del eje cartesiano. Este botón de la paleta de herramientas sólo se puede utilizar para las representaciones en dos dimensiones.

Si se pulsa el segundo botón del ratón sobre los ejes se despliega un menú secundario que permite manejar algunas opciones interesantes. Véase *Figura 118*.

- **Reset to Original View:** permite regresar a los ejes a su posición inicial de visión establecida.

En el menú secundario "Pan Options", es decir, opciones de visión panorámica, se despliega un sub-menú con las siguientes opciones:

- **Unconstrained Pan:** permite cambiar simultáneamente el punto de vista horizontal y vertical sin necesidad de barras de desplazamiento.
- **Horizontal Pan:** permite cambiar sólo el punto de vista horizontal.
- **Vertical Pan:** permite cambiar sólo el punto de vista vertical.

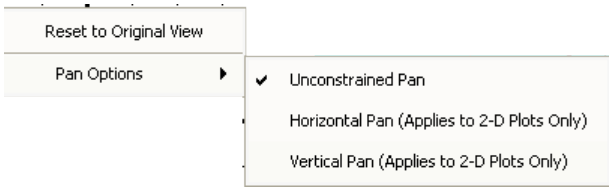


Figura 118. Menú contextual de la herramienta "Vista Panorámica"

Funcionamiento y aplicación de los menús de la interfaz gráfica

En primer lugar, se encuentra el menú *"Archivo"* que permite al usuario seleccionar entre cargar una sesión de trabajo anterior, guardar la sesión de trabajo realizada hasta el momento o abandonar la aplicación. Véase *Figura 119*.

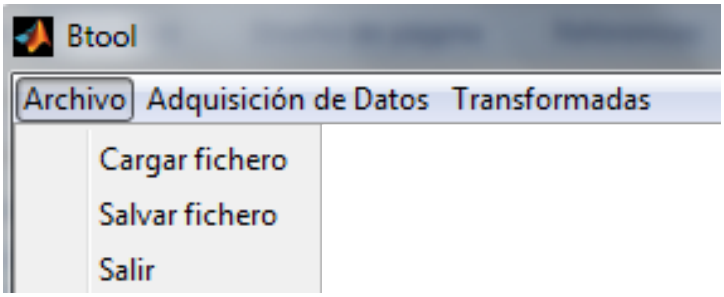


Figura 119. Opciones del menú "Archivo" de la interfaz gráfica "Btool.m"

En segundo lugar, se encuentra el menú *"Adquisición de Datos"*. La función de este menú es permitir al usuario elegir entre los tres modos de trabajo disponibles. Para poder acceder al modo "KEITHLEY" se necesita tener la correspondiente tarjeta de adquisición de datos conectada al PC. Para poder acceder al modo "Micrófono" se necesita tener conectado algún dispositivo a la clavija del micrófono. El modo "Simulación" se accede sin realizar ninguna conexión o porque no se haya podido acceder a otro modo de trabajo. Véase *Figura 120*.

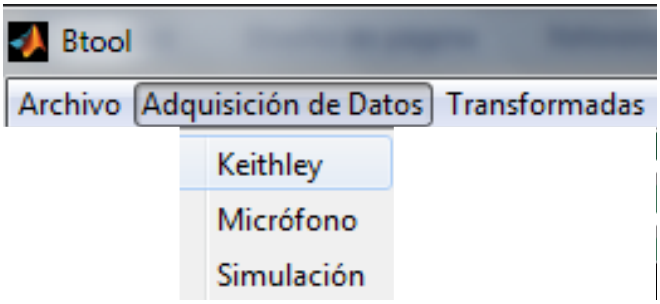


Figura 120. Opciones del menú "Adquisición de Datos" de la interfaz "Btool.m"

Por último, se encuentra el menú *"Transformadas"*. Este menú está compuesto por tres opciones que permiten al usuario realizar un tratamiento de señal diferente en función de las necesidades. Véase *Figura 121.*

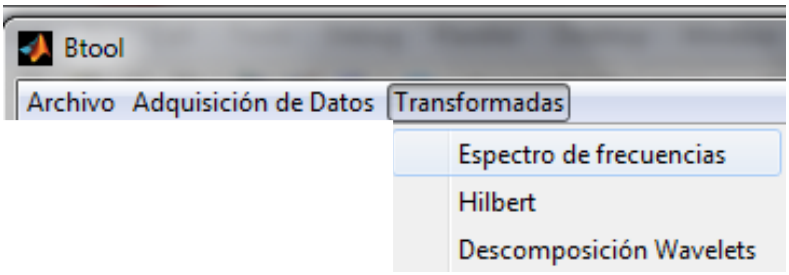


Figura 121. Opciones del menú "Transformadas" de la interfaz "Btool.m"

El tratamiento de señal mediante cualquiera de los tres tipos de transformadas sólo se puede realizar de una única señal, es decir, de un único canal seleccionado. Para poder elegir el canal que queremos seleccionar se debe elegir el canal utilizando en *"pop up menu"* de la interfaz gráfica de *"Btool.m"*. Véase *Figura 122.*

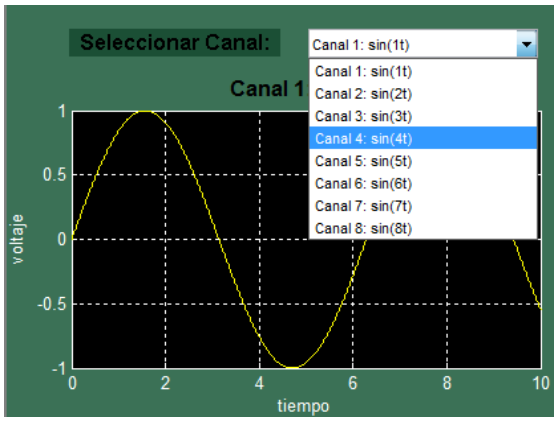


Figura 122. "Pop up menú" de la interfaz gráfica de "Btool.m"

5.4 Ejemplo de uso, resultados experimentales

Esta sección está dedicada a abordar diferentes casos prácticos reales o simulados para que el usuario que controle la interfaz gráfica disponga de un conocimiento rápido de cada modo de trabajo y tenga un criterio de selección de cada uno de ellos dependiendo de las circunstancias.

5.4.1 Modo Simulación

Se utiliza únicamente para un manejo básico, formación y entrenamiento de los operarios que no disponen de experiencia con la interfaz gráfica.

1. Ejecución de la interfaz gráfica situándose en el fichero *"Btool.m"* y pulsando el botón *"Run"*. 

2. Desplegar el menú contextual "Adquisición de Datos" y pulsar el menú *"Simulación"*. Véase *Figura 123*.

Figura 123. Modo Simulación

3. En el panel *"Seleccionar Canal"* marcar con un clic aquellos canales que se deseen representar de los ocho canales predefinidos. Véase *Figura 124*.

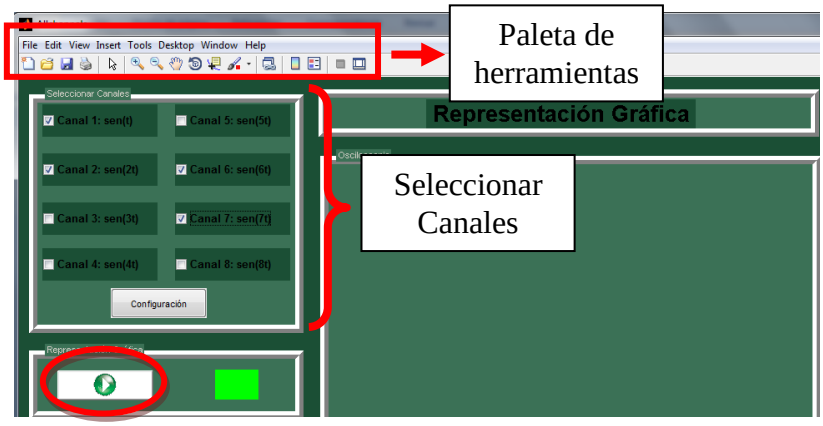


Figura 124. Selección de canales prefinidos y botón representación

4. Representación de los canales seleccionados pulsado el botón "Play" contenido en el panel "Representación Gráfica". Véase *Figura 124*.

5. Acondicionamiento de las señales representadas utilizando la *"Paleta de herramientas"* o el panel *"Escalado Ejes"* para adaptar las ondas a nuestra conveniencia. Véase *Figura 125*.

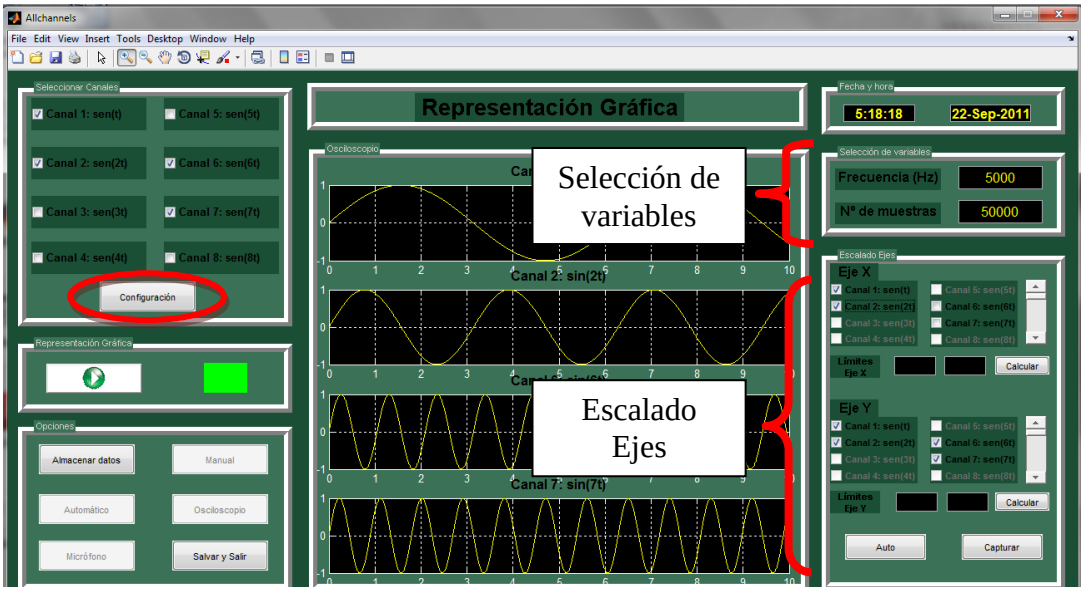


Figura 125. Escalamiento y tratamiento de señal en "Allchannels.m"

6. Modificación de la frecuencia y el número de muestras utilizando el panel *"Selección de variables"* y modificación de los nombres de las ondas pulsando el botón *"Configuración"* del panel *"Seleccionar Canal"*. Véase *Figura 125*.

7. Registrar los datos de las representaciones de onda en un fichero de texto pulsando el botón "Almacenar datos" y abandonar la ventana actual para regresar a la ventana *"Btool"* pulsando el botón "Salvar y Salir" ambos contenidos en el panel "Opciones". Véase *Figura 125*.

8. Desplegar el menú *"Transformadas"* para aplicar las diferentes transformaciones a las señales representadas. El tipo de transformación seleccionada se realizará únicamente a la señal representada en el menú desplegable. Véase *Figura 126* y *Figura 127*.

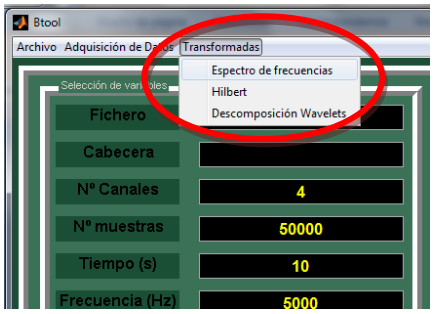


Figura 126. Menú Transformadas

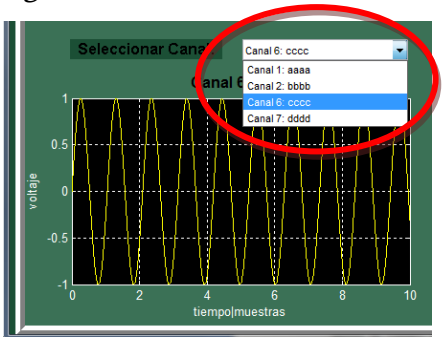


Figura 127. Menú desplegable

5.4.2 Modo Real

Se utiliza para la captación de señales reales por medio de la tarjeta de adquisición de datos *"KEITHLEY"* o gracias a la tarjeta de sonido integrada en el ordenador. Estos modos de adquisición de datos son totalmente independientes y cada uno tiene sus propias funcionalidades, propiedades y manera de trabajar.

Las señales reales que se desean capturar pueden provenir tanto de un generador de funciones como de un equipo industrial, sensor analógico u otros dispositivos.

Sin embargo, el generador de funciones se utiliza fundamentalmente para la programación, depuración y mejora del software de la interfaz gráfica que realiza la captura de datos. Por este motivo, se sabe tanto la frecuencia como la amplitud de las señales que se van a capturar con la tarjeta de adquisición al programarlas previamente en el generador de funciones. Esto es de gran ayuda porque el operario sabe los resultados que debe obtener con antelación y de esta manera se asegura que la aplicación funcione correctamente.

Asimismo, todas las discordancias entre la señal generada y la señal adquirida serán síntoma de que la aplicación no está realizando la captura de datos convenientemente. Otra ventaja de este modo es que el operario puede practicar a configurar de forma diferente los parámetros de la tarjeta de adquisición de datos para comprobar los límites y limitaciones de ésta y adecuarse de una manera más fiable a su entorno de trabajo.

Por tanto, se van a explicar detalladamente los diferentes modos de trabajo utilizando el generador de funciones para demostrar el correcto funcionamiento de la aplicación. La división de las funciones de trabajo para cada tarjeta de adquisición de datos es la siguiente:

- Tarjeta *"KEITHLEY":*
  - Función *"Osciloscopio".*
  - Función *"Automática".*
  - Función "*Manual".*
- Tarjeta de sonido:
  - Función *"Micrófono".*

Función Osciloscopio

Esta función permite representar hasta ocho canales simultáneamente en tiempo real, realizar cambios de escala para cada uno de los canales y finalmente almacenar la información de todos los canales con un número determinado de muestras en un fichero de texto.

Esta función es ideal para poder observar diversos canales y su evolución en el tiempo, poder compararlos entre ellos y realizar la captura de éstos en el momento más conveniente. A continuación, se detallará los pasos principales a seguir para realizar una captura de datos ordinaria:

1. Pulsar el botón *"Osciloscopio"* en la ventana de *"Allchannels.m"* y seleccionar los diferentes canales que se quieran representar en la ventana *"Tarjeta_adquisicion.m"*. Véase *Figura 128*.

2. Cargar la configuración automática recomendada pulsando el botón *"Auto"* y finalmente pulsar el botón *"Aceptar"* para confirmar la configuración de la tarjeta de adquisición de datos. Véase *Figura 128*.

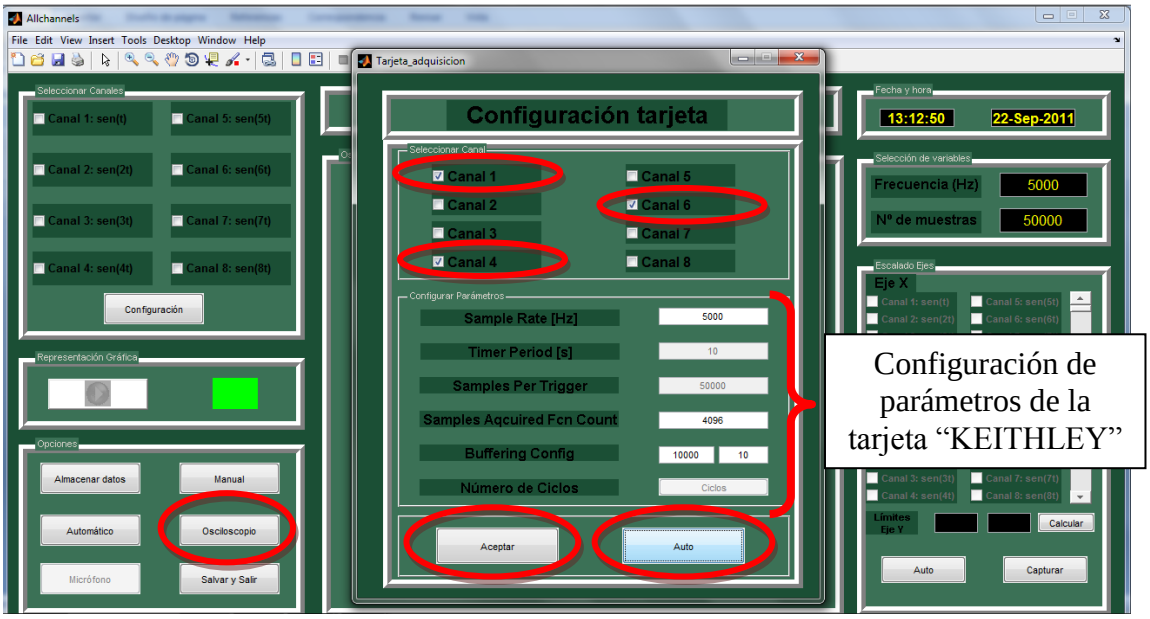


Figura 128. Configuración de la tarjeta de adquisición modo "Osciloscopio"

3. Pulsar el botón *"Trigger"* en la pantalla emergente *"Oscilloscope"* y se representarán en tiempo real las tres señales de entrada seleccionadas. Realizar un escalamiento vertical y horizontal de las señales para mejorar su representación gráfica mediante las ruletas que contiene el panel de *"Channel Scaling"*. Véase *Figura 129.*

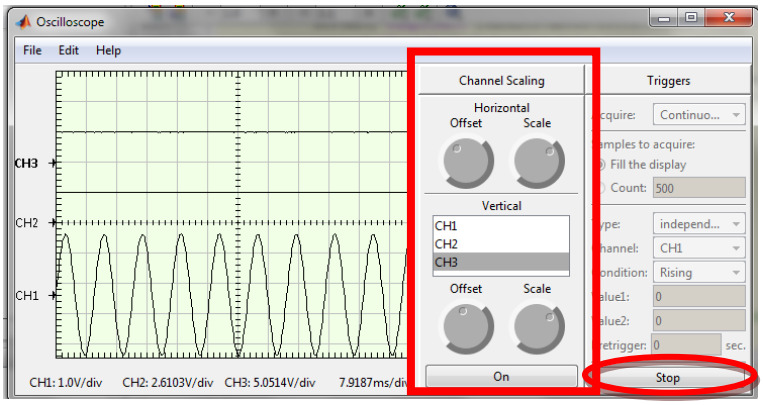


Figura 129. Representación y escalamiento de señales en osciloscopio

4. Seleccionar el menú *"File->Export->Channels..."* para realizar un almacenamiento de los datos en un fichero de texto. Para ello, seleccionar en el menú desplegable *"MAT-File"*, seleccionar la opción *"Count"* y escoger el número de muestras que se desean almacenar y por último elegir el nombre de las variables en la columna *"Variable Name"*. Pulsar el botón *"Export"* y *"Close"*. Véase *Figura 130*.

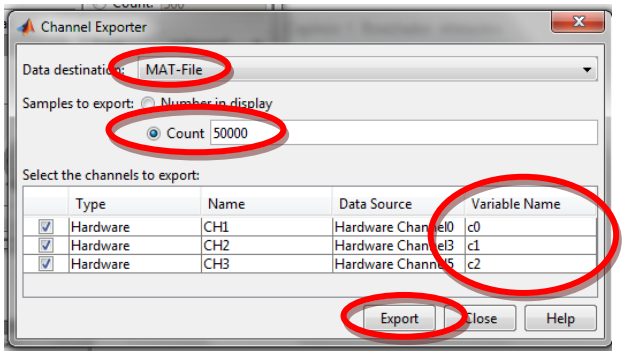


Figura 130. Ventana de exportación de datos de la función osciloscopio

5. Cerrar la ventana *"Oscilloscope"* y realizar la carga del fichero almacenado en el paso anterior. Véase *Figura 131*.

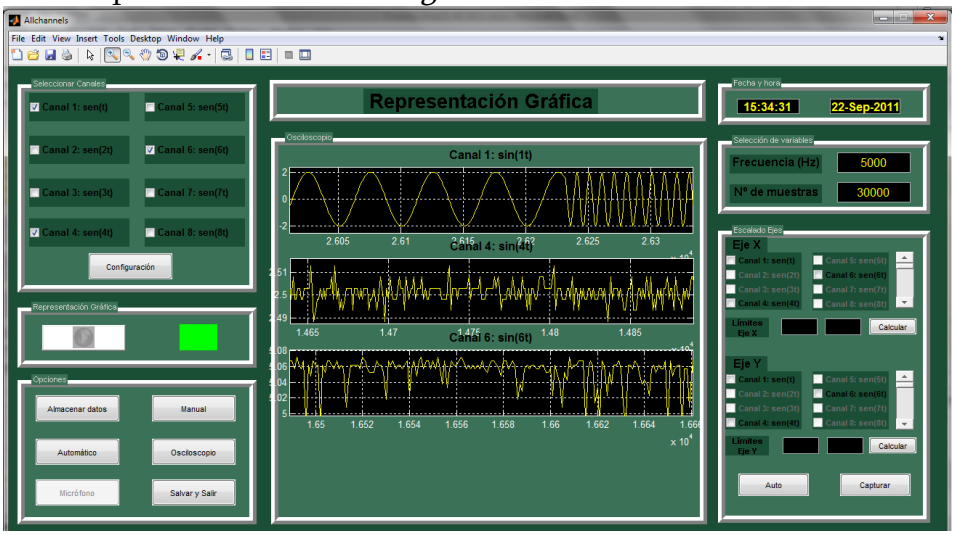


Figura 131. Representación de las señales capturadas por la función osciloscopio

Función Automática

Esta función permite almacenar hasta ocho canales durante un tiempo determinado que depende de la frecuencia de muestreo y las muestras totales a adquirir. No permite la representación gráfica en tiempo real de los canales seleccionados sino que el usuario sólo puede esperar a que termine el almacenamiento de todos los datos.

El objetivo de esta función de trabajo es dejar al software que trabaje adquiriendo información realizando repeticiones continuas de un mismo ensayo de trabajo para posteriormente comparar los datos obtenidos y sacar conclusiones. Para ello, se ha diseñado una interfaz que permite introducir al usuario el número de ciclos a realizar y el intervalo de tiempo de espera entre dichos ciclos. Estos ciclos se almacenan en la carpeta de registro correspondiente con la fecha y hora como nombre para tener una buena organización. A continuación, se detallará los pasos principales a seguir para realizar una captura automática de datos ordinaria:

1. Pulsar el botón *"Automático"* en la ventana de *"Allchannels.m"* y seleccionar los diferentes canales que se quieran representar en la ventana *"Tarjeta_adquisicion.m"*.

2. Cargar la configuración automática recomendada pulsando el botón *"Auto"* y finalmente pulsar el botón *"Aceptar"* para confirmar la configuración de la tarjeta de adquisición de datos. Véase *Figura 132*.

3. Pulsar el botón "Ciclos" para programar el número de repeticiones cíclicas a realizar del ensayo y para determinar el número de segundos entre cada repetición. Véase *Figura 133*.

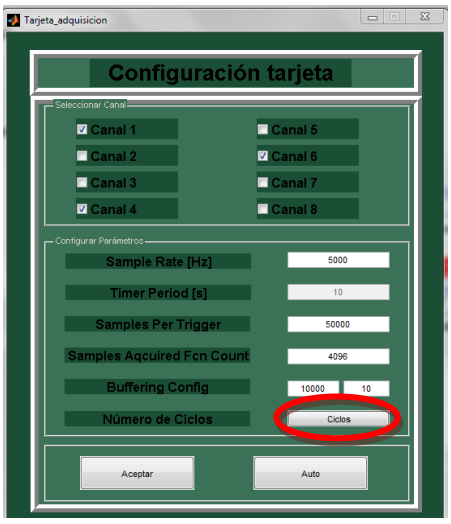


Figura 132. Configuración de la tarjeta de adquisición en modo "Automático"

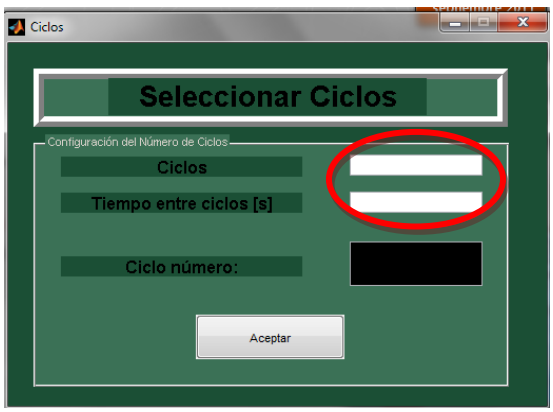


Figura 133. Configuración "Ciclos"

4. Según se termine el almacenamiento, se mostrará un contador de cada ciclo que se irá refrescando continuamente en la ventana *"Ciclos"* indicando el número de ciclos que ha almacenado. Véase *Figura 134*.

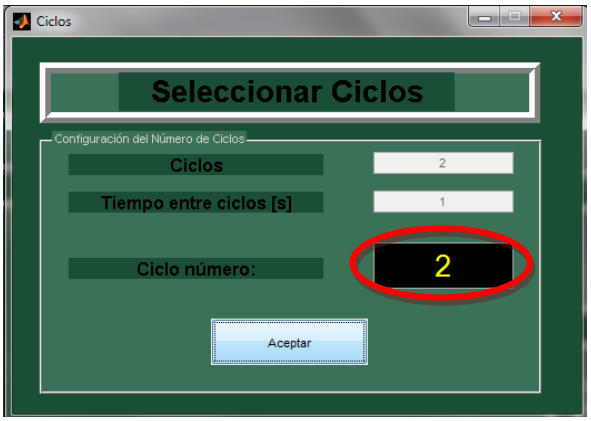


Figura 134. Contador de ciclos

5. Una vez almacenados todos los ciclos establecidos en la carpeta de registro correspondiente *"Registros_automaticos"*, se representará gráficamente en la ventana *"Allchannels.m"* el último ciclo capturado.

Función Manual

Esta función permite representar en tiempo real un único canal, y una vez que el usuario estime que es necesario comenzar a registrar los datos debido alguna incidencia, puede almacenar éstos durante el tiempo que dure el ensayo práctico.

El objetivo de dicha función es analizar en tiempo real un único proceso mediante la representación dinámica de su señal y si se detectan alteraciones en el funcionamiento normal, realizar la captura manual de datos a partir de dicho momento para su el posterior análisis de las causas de los posibles errores o defectos que se están produciendo.

Una vez terminada la adquisición y captura de datos se representa en la ventana *"Allchannels.m"* en el primer canal toda la señal desde el inicio de la aplicación y en el segundo canal se representa sólo a partir del momento en que el usuario a pulsado el botón de capturar los datos manualmente.

A continuación, se detallará los pasos principales a seguir para realizar una captura manual de datos ordinaria:

1. Pulsar el botón *"Manual"* en la ventana de *"Allchannels.m"* y seleccionar el único canal que se quiera representar en la ventana *"Tarjeta_adquisicion.m"*.

2. Cargar la configuración automática recomendada pulsando el botón *"Auto"* y finalmente pulsar el botón *"Aceptar"* para confirmar la configuración de la tarjeta de adquisición de datos.

3. Una vez cargada la ventana *"Captura_manual"* pulsar el botón *"Start"* para comenzar a monitorizar la señal entrante en la tarjeta de adquisición de datos. Véase *Figura 135*.

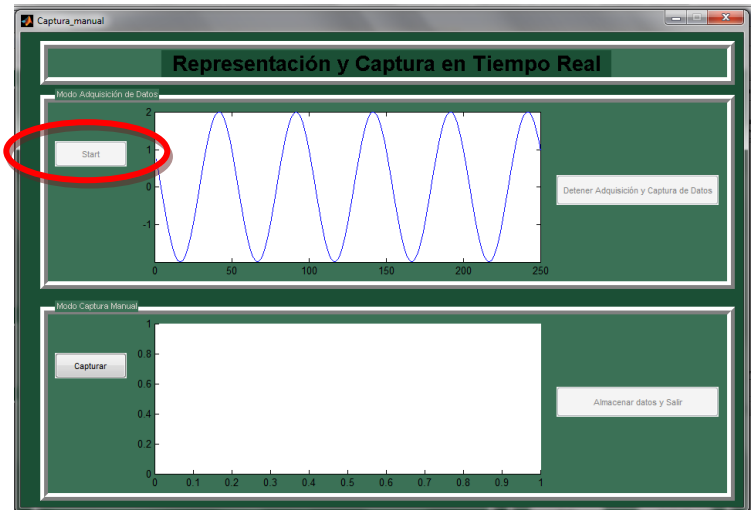


Figura 135. Monitorización de la señal entrante en el modo "Manual"

4. Cuando se considere necesario o una vez detectada la anomalía del ensayo, se pulsa el botón *"Capturar"* para comenzar a adquirir en el segundo canal datos a partir de ese momento.

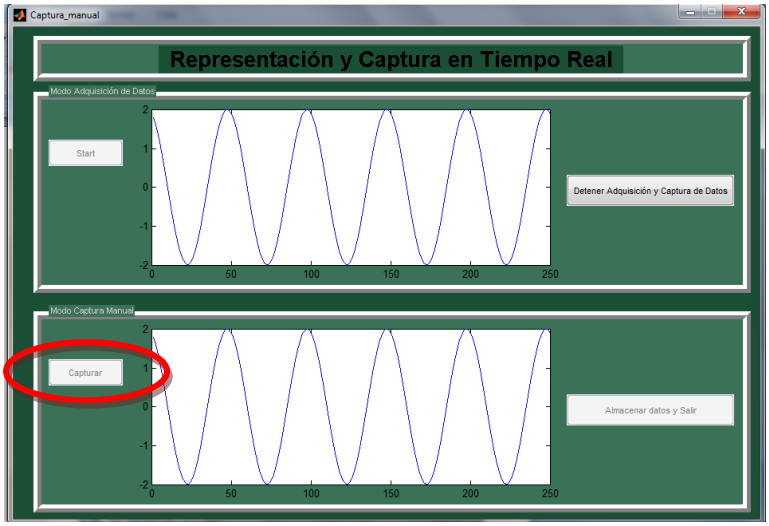


Figura 136. Captura de la señal entrante en el modo "Manual"

5. Se pulsa el botón *"Detener adquisición y Captura de datos"* y el botón de *"Almacenar datos y Salir"* para volver con todos los datos capturados a la ventana de *"Allchannels"* donde se realizará una representación gráfica de ambos canales. Véase *Figura 137*.

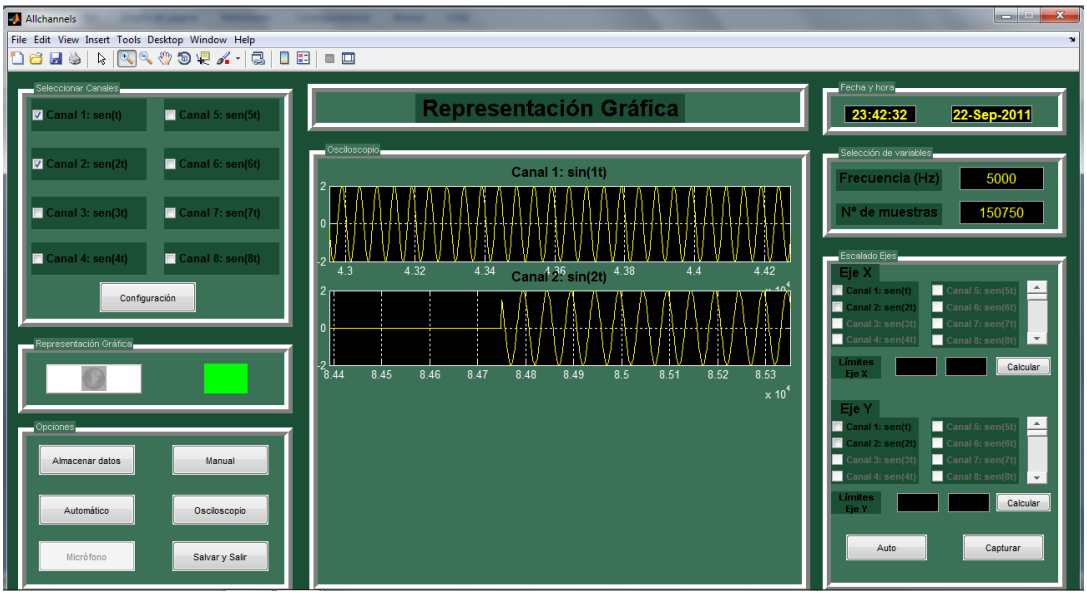


Figura 137. Representación de las señales capturadas por la función "Manual"

Función Micrófono

Esta función permite capturar mediante la tarjeta de sonido integrada en el ordenador datos de entrada a través de dos canales. Tiene la misma funcionalidad que la tarjeta de adquisición de datos *"KEITHLEY"* con el inconveniente que sólo tiene dos canales y la ventaja de que todos los ordenadores disponen de una tarjeta de sonido integrada que permite realizar capturas de datos sin necesidad de otros dispositivos.

Otra ventaja de este modo de trabajo es que se puede conectar a la clavija del micrófono cualquier dispositivo que se adapte al él y sea capaz de adquirir datos e información. Por ejemplo, el caso de los acelerómetros.

Sin embargo, la configuración de la tarjeta de sonido del ordenador es diferente que la configuración de la tarjeta de adquisición de datos *"KEITHLEY"*. Además no se dispone de ningún generador de señales que se pueda conectar a la clavija del micrófono que certifique que la captura de datos se realiza correctamente.

A continuación, se detallará los pasos principales a seguir para realizar una captura manual de datos ordinaria:

1. Pulsar el botón *"Micrófono"* en la ventana de *"Allchannels.m"* y se seleccionan automáticamente los dos únicos canales que se pueden representar en la ventana *"Tarjeta_adquisicion.m"*. Véase *Figura 138*.

2. Cargar la configuración automática recomendada pulsando el botón *"Auto"* y finalmente pulsar el botón *"Aceptar"* para confirmar la configuración de la tarjeta de adquisición de datos. Véase *Figura 138*.

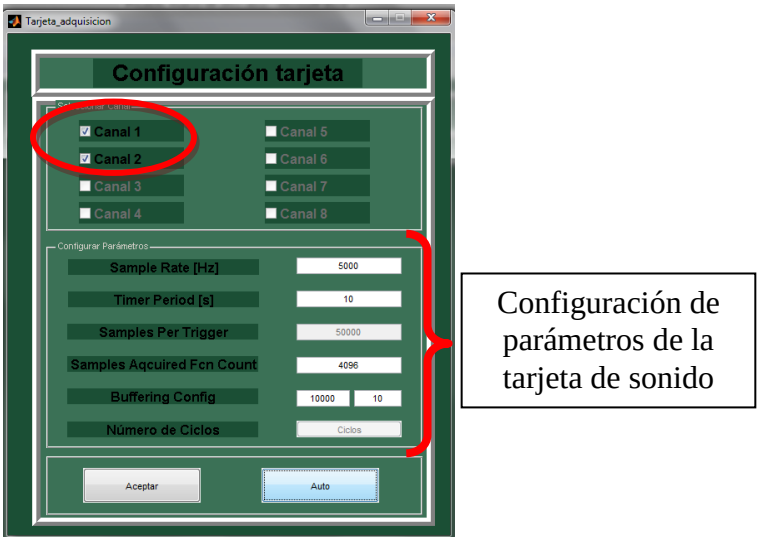


Figura 138. Configuración de la tarjeta del sonido en la función "Micrófono"

3. Una vez introducidos los parámetros de configuración de la tarjeta de sonido aparece una ventana donde se representa tanto la adquisición de datos en tiempo real como su representación en función de su magnitud [dB] y frecuencia [Hz]. Véase *Figura 139*.

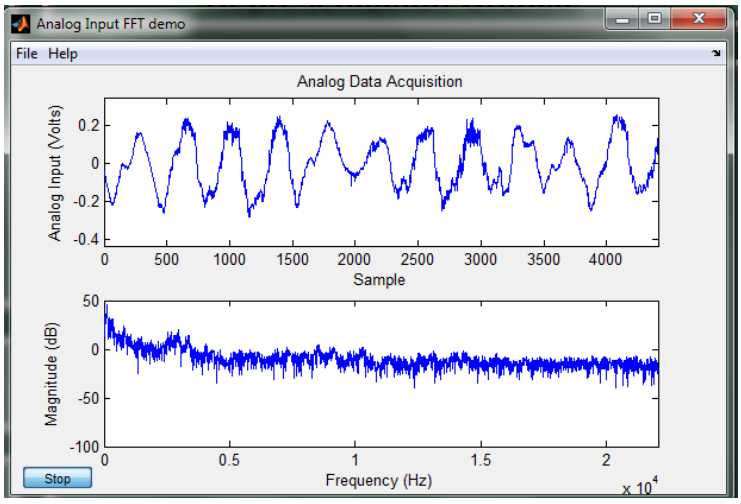


Figura 139. Representación gráfica de la captura de datos en modo "Micrófono"

4. Después de esperar el tiempo de adquisición determinado por el usuario, la ventana emergente donde se representan gráficamente los datos se cierra automáticamente y se representan todos los datos almacenados por los dos canales en la ventana "Allchannels.m". Véase *Figura 140*.

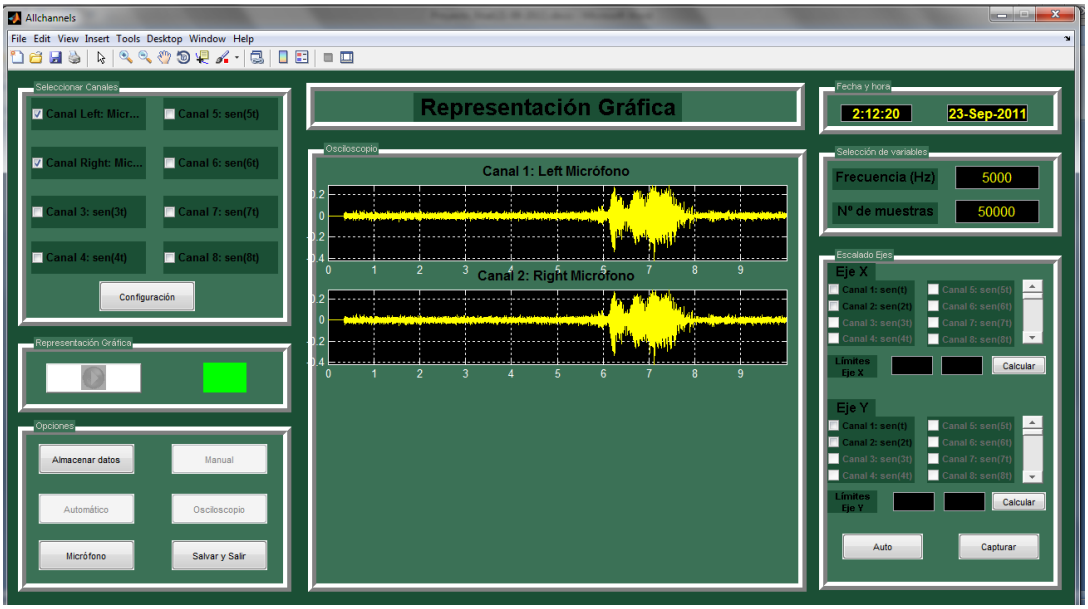


Figura 140. Representación de las señales capturadas por la función "Micrófono"

A continuación, se muestran las carpetas y funciones que forman la aplicación de *"Btool"* en la *Figura 141*. En la *Figura 142* se muestran los registros de texto almacenados en los diferentes modos de trabajo.

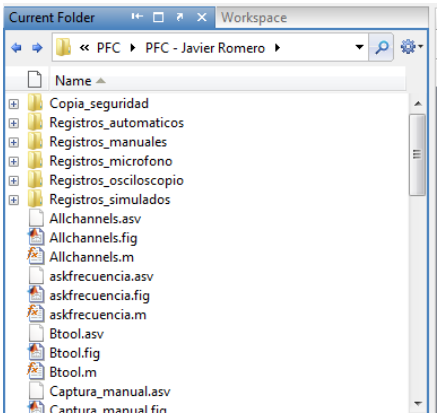


Figura 141. Registros y funciones

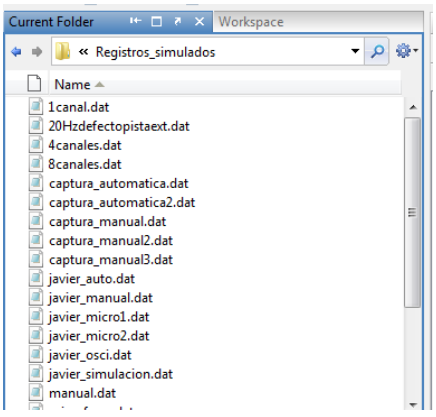


Figura 142. Ficheros de texto (.dat)*

Por último, es conveniente destacar que todas las herramientas de personalización y tratamiento de las señales que se han explicado para el modo de "Simulación" se pueden utilizar en cualquier modo de funcionamiento.

5.5 Consideraciones y advertencias

Antes de comenzar a utilizar la interfaz gráfica *"Btool"* para la adquisición de señales es recomendable tener en cuenta las siguientes consideraciones y advertencias para que ésta funcione correctamente y así evitar posibles errores que provoquen la salida de la aplicación y la pérdida de la información almacenada en la sesión de trabajo iniciada.

1. El modo *"Manual"* sólo se puede rastrear un único canal ya que se realiza un adquisición de datos en tiempo real y ofrece la posibilidad al usuario de capturar la información a partir del momento que le resulte oportuno.

2. El modo *"Osciloscopio"* y el modo *"Automático"* permiten muestrear a la vez desde uno hasta ocho canales, pudiendo representarlos y almacenarlos todos a la vez para poder realizar comparaciones entre ellos.

3. El modo *"Micrófono"* sólo permite muestrear dos canales, es decir, los correspondientes a la tarjeta de sonido del altavoz izquierdo y derecho. Modo muy útil para realizar capturas de señal sin necesidad de tener ninguna tarjeta de adquisición de datos.

4. No se puede utilizar una frecuencia de muestreo para la tarjeta *"KEITHLEY"* o la tarjeta del micrófono menor o igual que la frecuencia de la señal de trabajo. La frecuencia de muestreo tiene que ser al menos cien veces mayor que la frecuencia de trabajo para poder realizar un buen muestreo y recuperar perfectamente la señal entrante.

5. Hay que introducir correctamente y con conocimiento de causa los parámetros de configuración tanto de la tarjeta *"KEITHLEY"* como de la tarjeta de sonido, ya que se pueden producir errores o saturación provocando un fallo en el software.

6. Es preferible realizar los escalamientos de las señales en la interfaz gráfica de "Allchannels.m" mediante la lupa de la paleta de herramientas o mediante los *"edit text"*. Evitar el uso de los *"sliders"* excepto en el modo "Simulación".

7. No se pueden realizar tratamientos de señal en *"Allchannels.m"* de los ficheros de texto almacenados en las carpetas de registro. La única finalidad de función *"cargar.m"* es volver a representar señales capturadas en sesiones anteriores de trabajo para poder realizar un estudio de sus transformaciones de "*Fourier, Hilbert o Descomposción de Wavelets*".

6. Conclusiones

6.1 Conclusiones

En este apartado, se hará una recapitulación de todos los objetivos propuestos al principio del presente Proyecto de Fin de Carrera y se contrastarán los objetivos iniciales con los resultados finales obtenidos. Además se comentarán las mejoras e innovaciones que han surgido durante el desarrollo de estos objetivos.

Se ha conseguido llevar a cabo el objetivo principal de desarrollar una interfaz gráfica externa que permita la adquisición, almacenamiento, procesamiento y monitorización de señales para la detección de fallos mecánicos en los rodamientos y ejes.

Se han realizado mejoras tanto en la visualización de la interfaces gráficas externamente como en la programación interna del software, dotándole de una mayor funcionalidad y facilidad de uso.

Además, se han añadido menús interactivos que permiten al usuario seleccionar el modo de trabajo que más se adapte a las necesidades de trabajo en cada circunstancia y disponer de un modo de trabajo con datos simulados para no necesitar realizar pruebas reales con el software en cada sesión.

Se ha ampliado la funcionalidad de la interfaz gráfica gracias a la inserción de paletas de herramientas gráficas y la creación de botones funcionales para que el usuario pueda interactuar. Se ha conseguido implementar un panel propio para realizar un escalamiento adecuado de las señales adquiridas y gracias a éste poder apreciar con mayor precisión los defectos en los ejes y rodamientos provocados en las diferentes sesiones de trabajo.

Se ha realizado el almacenaje de todos los valores paramétricos en un archivo de texto para que después de finalizar la aplicación se puedan volver a cargar dichos datos y trabajar en sesiones posteriores de trabajo. Además, de esta manera, se pueden comparar unos registros con otros para analizar la fiabilidad de la toma de datos realizada.

Todo ello, ha contribuido para que la interfaz gráfica se convierta en una aplicación mucho más interactiva, de fácil de utilización y que permita al usuario un gran abanico de posibilidades para realizar las adquisiciones de datos de la forma más optimizada e intuitiva posible. Por tanto, la interfaz gráfica se ha convertido en un sistema fundamentalmente visual e intuitivo, para que el usuario que esté controlando la aplicación, pueda tener en todo momento un conocimiento rápido, preciso y en tiempo de real de cómo va evolucionando las señales generadas por el sistema para su correcto control y prevención de riesgos.

La interfaz gráfica dispone del modo de trabajo simulación que permite a los usuarios noveles realizar prácticas y entrenamientos para dominar las funcionalidades básicas de la interfaz. Además, no necesita la conexión de ningún dispositivo ni estar en un ambiente de trabajo específico por lo que se ha convertido en la perfecta herramienta o consola virtual para la formación de futuros controladores de la aplicación.

Las comunicaciones entre los diferentes módulos implicados en el sistema de han sido correctamente configuradas y sincronizadas para realizar automáticamente todo el flujo de intercambios de variables sin producirse ningún fallo de lectura o escritura de datos y permitir a los usuarios disponer ordenadamente de todos los ficheros de texto.

El tiempo de ejecución del software se ha depurado totalmente para que el muestreo de las señales y los intercambios de datos lleguen rápidamente a la interfaz gráfica externa para que ésta pueda monitorizar dichas capturas de datos prácticamente en tiempo real.

6.2 Trabajos futuros y ampliaciones

Los resultados obtenidos han cumplido con rigor con la expectativas marcadas por los objetivos iniciales del Proyecto de Fin de Carrera pero lo cierto es que es difícil poner fin a un proyecto de esta categoría, pues siempre hay nuevas mejoras que introducir, que conviertan a la interfaz gráfica en una aplicación un poco más real y práctica.

A continuación, se exponen algunas mejoras que podrían ser tenidas en cuenta para una futura optimización de este proyecto:

- Programar un sistema de alertas en el software de la interfaz gráfica para evitar posibles situaciones de peligro. Es decir, que avise al usuario que controla el proceso de adquisición de datos de los ejes y rodamientos cuando los valores capturados estén fuera de los límites establecidos.

- Optimizar el software para conseguir que el sistema de adquisición de datos y representación en tiempo real refresque a una velocidad mayor para poder detectar variaciones más rápidamente.

- Transformar el software actual basado en una arquitectura mono-hilo, en una aplicación multi-hilo, lo cual produciría una significativa mejora en el rendimiento de toda la aplicación, ya que en el modelo actual los procesos se ejecutan secuencialmente con toda la latencia y el retardo que esto conlleva, y con una nueva versión multi-hilo todos los procesos se ejecutarían concurrentemente, lo cual dotaría al sistema de mucha más fluidez a la hora de trabajar con él.

- Trasladar todo el software generado con MATLAB a otros lenguajes gráficos más potentes y con muchas más posibilidades, por ejemplo, “Delphi” o “Visual Basic”.

En definitiva, estas propuestas buscan realizar una aplicación más sólida, versátil y funcional tanto en las posibilidades que ofrezca, como en el modo en que éstas se presenten al usuario, con la intención de proporcionar una herramienta de trabajo y aprendizaje más novedosa, que facilite la detección de fallos en ejes y rodamientos de una manera más automatizada y optimizada.

Bibliografía

MANUALES

[1] GARCÍA DE JALÓN, Javier; IGNACIO RODRÍGUEZ, José; VIDAL, Jesús. "Aprenda Matlab 7.0 como si estuviera en primero". Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Madrid. Diciembre de 2005.
http://mat21.etsii.upm.es/ayudainf/aprendainf/Matlab70/matlab70primero.pdf

[2] CASADO FERNÁNDEZ, MªCristina. "Manual Básico de Matlab". Servicios Informáticos U.C.M. Apoyo a Investigación y Docencia. Universidad Complutense de Madrid.
http://www.sisoft.ucm.es/Manuales/MATLAB_r2006b.pdf

[3] BARRAGÁN GUERRERO, Diego Orlando. "Manual de interfaz gráfica de usuario en Matlab. Parte I". Ecuador.
http://www.matpic.com/MATLAB/MATLAB_GUIDE.html

[4] MEIROVITCH, L. "Elements of vibration analysis". Second edition. Mc Graw-Hill, (1986).

[5] ESTUPIÑAN, E., "Diagnóstico de fallas en máquinas de baja velocidad utilizando análisis de vibraciones". Tésis para la obtención del máster en Ciencias de la Ingeniería con Mención en Ingeniería Mecánica de la universidad de Concepción, Chile (2001).

[6] DUQUE PÉREZ, O., "Técnicas de mantenimiento preventivo". Ed. Abecedario,(2009).

[7] ERICSON, S., GRIP, N., JOHANSSON, E., PERSON, L., SJÖBERG, R., y STRÖMBERG, J. "Automatic detection of local bearing defects in rotating machines. Part.1". Departamento de matemáticas de la universidad tecnológica de Lula, (2001).

[8] WHITE, G. "Introducción a la vibración de máquinas". Part number 8569, versión 1.75, DLI Engineering, (1995).

[9] BRAUN, S. "Analysis of Roller/Ball Bearing Vibrations". ASME Publication (1979).

PÁGINAS WEB

[10] Keithley Instruments, Inc. Actualización 2011.
http://www.keithley.com/products/data/multifunction/usb/?mn=KUSB-3100

[11] MATLAB, MathWorks. R2011b Documentación - Data Acquisition Toolbox Actualización 2011
http://www.mathworks.es/help/toolbox/daq/f13-25414.html

[12] Data Translation, Inc. USB Data Adquisition. Actualización 2011.
http://www.datatranslation.com

[13] RENOVETEC, "Ingeniería de mantenimiento". Colección mantenimiento industrial. Disponible en Internet.
http://www.renovetec.com

PROYECTOS

[14] SANZ ARRANZ, Álvaro. “Diseño de una herramienta con MATLAB para la adquisición y procesamiento de señales. Aplicación a sistema de detección de fallos de rodamientos”- Tutor: Ramón Ignacio Barber Castaño y Cristina Castejón Sisamón. Universidad Carlos III de Madrid, Departamento de Ingeniería de Sistemas y Automática. 2010.

[15] ADRADOS SOMOLINOS, Alberto. “Análisis de sistemas de clasificación de defectos en ejes”. Tutor: Cristina Castejón Sisamón. Universidad Carlos III de Madrid, Departamento de Ingeniería Mecánica.

[16] ROMERO CARRASCO, Javier. “Interfaz gráfica para la monitorización del guiado de máquinas tuneladoras”. Tutor: Alberto Jardón Huete. Cotutor: Carlos González de la Vega. Universidad Carlos III de Madrid, Departamento de Ingeniería de Sistemas y Automática 2009.

Anexos

ANEXO A. Hoja de características de la tarjeta KEITHLEY

KUSB-3100 Series USB-Based Data Acquisition Modules

externally on the user connections. You can also set the duty cycle, frequency, and output polarity of the output pulse from the user counter/timers.

User Connections

A single USB cable, shipped with each KUSB-3100, -3102A, -3108, and -3160 module, provides both power and signal connections from your PC. No external power supply or battery is required. The KUSB-3116 includes a USB cable and +5V 2A power supply and power cable.

Signal connections are made directly to each module using either screw terminals, BNC connectors, or 37-pin D-type connectors located on the module itself. The KUSB-3102A and -3108 provide removable screw terminal blocks. Pin assignments are clearly marked on the module labels for quick setup.

KUSB-3100 Series Specifications

ANALOG INPUT SPECIFICATIONS

	KUSB-3100	KUSB-3102A	KUSB-3108	KUSB-3116
Number of Analog Input Channels	8 SE	16 SE/pseudo-DI, 8 DI	16 SE/pseudo-DI, 8 DI (7 T/C, 1 CJC)	16 SE/8 DI
Resolution	12-bit	12-bit	16-bit	16-bit
Channel-Gain List	16 locations	32 locations	32 locations	1024 locations
Input FIFO Size	2000[3] samples	512 samples	2048 samples	2048 samples
Gains	1, 2, 4, 8	1, 2, 4, 8	1, 10, 100, 500	1, 2, 4, 8
Input Range				
Bipolar	±10, 5, 2.5, 1.25 V	±10, 5, 2.5, 1.25 V	±10, 1, 0.1, 0.02 V	±10, 5, 2.5, 1.25 V
Unipolar		0–10, 5, 2.5, 1.25 V		
System Accuracy to % of FSR (averaged over 50 readings)	0.04% @ Gain = 1 0.06% @ Gain = 2 0.08% @ Gain = 4 0.15% @ Gain = 8	0.03% @ Gain = 1 0.04% @ Gain = 2 0.05% @ Gain = 4 0.05% @ Gain = 8	0.01% @ Gain = 1 0.02% @ Gain = 10 0.03% @ Gain = 100 0.04% @ Gain = 500	See note 2
Nonlinearity	0.05%	±1 LSB	±4 LSB	<0.5 LSB
Differential Nonlinearity	±0.5 LSB	±0.5 LSB (no missing codes)	±1.2 LSB (no missing codes)	0.5 LSB
Drift				
Zero	±100 µV	±30 µV + (20 µV · Gain)/°C	±25 µV + (5 µV · Gain)/°C[4]	±10 µV/°C
Gain	±100 ppm	±30 ppm/°C	±20 ppm/°C	±30 ppm/°C
Differential Linearity	Monotonic			±2 ppm/°C
Input Impedance[1]				
Off Channel	10 MΩ, 10 pF	100 MΩ, 10 pF	100 MΩ, 10 pF	100 MΩ, 10 pF
On Channel	10 MΩ, 100 pF	100 MΩ, 100pF	100 MΩ, 100 pF	100 MΩ, 100 pF
Input Bias Current	±10 nA	±20 nA	±10 nA	±20 nA
Maximum Input Voltage (without damage)				
Power On	±35 V	±40 V	±40 V	±35 V
Power Off	±20 V	±20 V	±20 V	±20 V
Common Mode Voltage	±11 V max. (operational)	±11 V max. (operational)	±11 V max. (operational)	±11 V max. (operational)
Common Mode Rejection	N/A	>74 dB	>74 dB	80 dB (gain = 1 @ 1 kΩ)
A/D Conversion Time	8 µs	6.6 µs	8 µs	2 µs
A/D Converter Noise	0.6 LSB rms	0.3 LSB rms	0.4 LSB rms	0.4 LSB rms
Channel Acquisition Time	20 µs (±0.5 LSB)	3 µs	6 µs (gain = 1) 250 µs (gain = 500)	1 µs (±0.5 LSB) typical
Channel-to-Channel Offset	0.1mV	±40 µV	±40 µV	±40 µV
Throughput	50 kS/s	100 kS/s	50 kS/s	500 kS/s (single channel), 500 kS/s + 0.05% (multiple channel)
CJC Voltage @ 25°C			0.25 V	
CJC Accuracy			1° from 5° to 45°C	
CJC Warm-up Time			10 to 20 minutes	

RECOMMENDED ACCESSORIES

KUSB-CABDIO	100-pin Cable for KUSB-STP100 and KUSB-3160
KUSB-STP100	Screw Terminal Panel for KUSB-3160

OPTIONAL ACCESSORIES

4801	48″ Low Noise Coax BNC to BNC Cable. For KUSB-3116
7051-2	2′ General Purpose BNC to BNC Cable. For KUSB-3116
7051-5	5′ General Purpose BNC to BNC Cable. For KUSB-3116
7051-10	10′ General Purpose BNC to BNC Cable. For KUSB-3116
7401	Type K thermocouple Wire Kit, 100 ft. For KUSB-3108
C1800	18″ Ribbon Cable with one 37-Pin D-type female connector and one 37-Pin D-type male connector. For KUSB-3116
C-1800/M	18″ Ribbon Cable with two 37-Pin D-type female connectors. For KUSB-3116
KUSB-BNC-DIN-KIT	DIN Rail Kit for KUSB-3116
KUSB-DIN-MOUNT-KIT	DIN Rail Mount Kit for KUSB-3102A, -3108, -3160
KUSB-ST	Extra Screw Terminal Blocks for KUSB-3102A and KUSB-3108
S1803	Shielded 4.5′ Cable with 37-Pin D-type female connectors. For KUSB-3116
S1805	Shielded 6.5′ Cable with 37-Pin D-type female connectors. For KUSB-3116
STA-U	Universal Screw Terminal Accessory for KUSB-3116

SYSTEM REQUIREMENTS:

PC with Pentium 233MHz processor minimum
256MB RAM or higher recommended
Windows 2000/XP/Vista operating system
USB ports – One or more (version 2.0 or 1.1) (USB version 2.0 is required to reach full speed capabilities of some modules.)
Super VGA (800 × 600) or higher resolution monitor
CD-ROM drives – one or more

Speed, resolution, and channel capacity to meet demanding measurements

DATA ACQUISITION PRODUCTS

1.888.KEITHLEY (U.S. only)
www.keithley.com

KEITHLEY
A GREATER MEASURE OF CONFIDENCE

KUSB-3100 Series specifications

DATA ACQUISITION PRODUCTS

KUSB-3100 Series

USB-Based Data Acquisition Modules

KUSB-3100 Series Specifications (continued)

ANALOG INPUT SPECIFICATIONS (continued)

	KUSB-3100	KUSB-3102A	KUSB-3108	KUSB-3116
Thermocouple Break Detection Current			50 nA (high-side differential) outputs full-scale for gains greater than 1, or 2.5 V for gains of 1	
Effective Number of Bits (ENOB)	10.5 bts typical	11.5 bits	14.1 bits	14.6 bits
Total Harmonic Distortion (THD)	<-70 dB typical	–80 dB typical	–90 dB typical	–90 dB typical
Channel Crosstalk	–74 dB @ 1 kHz	–80 dB @ 1 kHz	–80 dB @ 1 kHz	–80 dB @ 1 kHz
Maximum A/D Pacer Clock				
Single Analog Input Throughput	50 kHz	100 kS/s @ 0.03% accuracy	50 kS/s @ 0.01% accuracy at gain of 1–10	50 kS/s
Multiple Analog Input Throughput	50 kHz	100 kS/s @ 0.03% accuracy	50 kS/s @ 0.01% accuracy at gain of 1–10 10 kS/s @ 0.03% accuracy at gain of 100 2 kS/s @ 0.04% accuracy at gain of 500	50 kS/s @ ±0.05%
Minimum A/D Pacer Clock Throughput	0.75 S/s	0.75 S/s	0.75 S/s	0.00419 S/s
External A/D Sample Clock				
Minimum Pulse Width	200 ns (high) 200 ns (low)	600 ns (high) 600 ns (low)	600 ns (high) 600 ns (low)	25 ns (high) 25 ns (low)
Maximum Frequency (Analog Inputs)	50 kHz	100 kHz	50 kHz	500 kHz
Maximum Frequency (Digital Inputs Only)	N/A	Maximum A/D rate	Maximum A/D rate	Maximum A/D rate
External Digital (TTL) Trigger				
High Input Voltage	2.4 V min.	2.4 V min.	2.4 V min.	3.3 V min.
Low Input Voltage	0.8 V max.	0.8 V max.	0.8 V max.	0.8 V max.
Minimum Pulse Width	200 ns (high) 200 ns (low)	600 ns (high) 600 ns (low)	600 ns (high) 600 ns (low)	25 ns (high) 25 ns (low)

[1] Very high input impedance minimizes any source error.

[2] System accuracy (% of FSR) of KUSB-3116

	500kHz	400kHz	250kHz
Gain = 1	±0.05%	±0.03%	±0.01%
Gain = 2	±0.06%	±0.04%	±0.02%
Gain = 4	±0.07%	±0.05%	±0.02%
Gain = 8	±0.09%	±0.07%	±0.03%

[3] Total FIFO size used for both A/D and D/A on the module is 2k.

[4] This value is referenced to voltage entering the A/D converter. To reference this value to the original voltage signal, use {[±25μV + (5μV · Gain)]/Gain}/°C.

1.888.KEITHLEY (U.S. only)

www.keithley.com

KEITHLEY

A GREATER MEASURE OF CONFIDENCE

KUSB-3100 Series

USB-Based Data Acquisition Modules

ANALOG OUTPUT SPECIFICATIONS

	KUSB-3100	KUSB-3102A	KUSB-3108	KUSB-3116
Number of Analog Output Channels	2	2	2	4
Resolution	12-bit	12-bit	16-bit	16-bit
Output Range	±10 V	±5, ±10, 0–5, 0–10 V	±10 V	±10 V
Nonlinearity	0.05%	±1 LSB	±4 LSB	±1 LSB
Differential Nonlinearity	±1 LSB	±1 LSB	±1 LSB	±1 LSB
Differential Linearity	±1 LSB (monotonic)	±0.5 LSB (monotonic)	±1.0 LSB (monotonic)	±1 LSB (monotonic)
Error				
Gain	±0.2%	±2 LSB + reference	±6 LSB	Adjustable to 0
Zero	±4 mV	Software adjustable to 0	Software adjustable to 0	Adjustable to 0
Drift				
Zero (bipolar)	±100 µV/°C	±10 ppm of FSR/°C	±10 ppm of FSR/°C	±10 ppm of FSR/°C
Gain	±100 ppm	±30 ppm of FSR/°C	±30 ppm of FSR/°C	±30 ppm of FSR/°C
Throughput				
Single Value		50 Hz	System dependent	
Waveform Generation Mode	50 kHz			500 kS/s
Continuously Paced Analog Output Mode	50 kHz			500 kS/s
Current Output	+2 mA max. load	+5 mA max. load	+5 mA max. load	+5 mA max. load
Output Impedance	<0.2 Ω	0.3 Ω typical	0.3 Ω typical	0.1 Ω
Capacitive Drive Capability	1000 pF min.	0.001 µF (no oscillators)	0.001 µF (no oscillators)	0.004 µF
Protection	Short to ground	Short to analog common	Short to analog common	Short to analog common
Power-on Voltage	0V ±10 mV	0V ±10 mV	0V ±10 mV	0V ±10 mV
Setting Time to 0.01% of FSR	20 µs	50 µs, 20 V step; 10.0 µs, 100 mV step	50 µs, 20 V step; 10.0 µs, 100 mV step	5 µs, 10 V step; 4 µs, 100 mV step
Slew Rate	2 V/µs	2 V/µs	2 V/µs	10 V/µs

DIGITAL I/O SPECIFICATIONS

	KUSB-3100	KUSB-3102A	KUSB-3108	KUSB-3116	KUSB-3160[2]
Number of Digital I/O Lines	8 in, 8 out	8 in (Port A), 8 out (Port B)	8 in (Port A), 8 out (Port B), 1 dynamic digital output	16 in, 16 out, 1 dynamic digital output	8 bidirectional per port (Ports 0–7), 8 inputs per port (Ports 8–11)
Logic Family	TTL	TTL	TTL	LVTTL	TTL
Logic Sense	Positive true	Positive true	Positive true	Positive true	Positive true
Inputs					
Input Type	Level sensitive	Level sensitive	Level sensitive	Level sensitive	Level sensitive
Input Logic Load	1 TTL load	1 TTL load	1 TTL load	1 LVTTL load	1 TTL load
High Input Voltage	2.4 V min.	2.0 V min.	2.0 V min.	2.0 V min.	2.0 V min.
Low Input Voltage	0.8 V max.	0.8 V max.	0.8 V max.	0.8 V max.	0.8 V max.
Low Input Current	–0.4 mA max.	–3 µA	–3 µA	–0.4 mA max.	
High Input Current		3 µA	3 µA		
Pulse Width (min.)					66 ns high and low[1]
Internal Pacer Clock Rate (max.) (single digital channel)	N/A	Max. A/D rate	Max. A/D rate (Port A and dynamic digital output)	Max. A/D rate	N/A
Outputs					
Fan Out	12 mA	12 mA	12 mA	12 mA	12 mA
High Voltage Output	2.8 V min.	74 HCT 244 (TTL) 2.4 V min.	2.4 V min.	2.0 V min.	2.4 V min.
Low Voltage Output	0.6 V max.	0.5 V max.	0.5 V max.	0.8 V max.	0.5 V max.
High Output Current (Source)	2 mA	1 mA	–1 mA Port B 1 mA DDO	–12 mA	–15 mA
Low Output Current (Sink)	10 mA	12 mA	12 mA (Port B) or 2 mA (dynamic digital output)	12 mA	12 mA
Interrupt on Change	No	No	No	Yes	Yes
Clocked with Sample Clock	Yes	Yes	Yes	Yes	No

1 The minimum pulse width applies only to interrupt-on-change detection for Ports 1 and 2. Pulses less than the minimum may not be detected as a change. All diodes back EMF protected for inductive loads.

2 The mass termination connector is a 100-pin D, Robinson Nugent part #P50E-100P1-SR1-TG. The mating connector is a 100-pin Robinson Nugent part #P50E-100S-TG.

KUSB-3100 Series specifications

DATA ACQUISITION PRODUCTS

KUSB-3100 Series

USB-Based Data Acquisition Modules

COUNTER/TIMER SPECIFICATIONS

	KUSB-3100	KUSB-3102A	KUSB-3108	KUSB-3116
Number of Counter/Timers	1	2	2	5[1]
Resolution	32-bit	16-bit	16-bit	32-bit
Minimum Pulse Width (minimum amount of time for a C/T to recognize an input pulse)	200 ns	600 ns (high); 600 ns (low)	600 ns (high); 600 ns (low)	55.5 ns
Logic Family	TTL	TTL	TTL	LVTTL
Inputs	Level sensitive	Level sensitive	Level sensitive	Edge sensitive
Input Logic Load	1 TTL load	1 TTL load	1 TTL load	1 LVTTL load
High Input Voltage	2.4 V min.	2.4 V min.	2.4 V min.	2.0 V max.
Low Input Voltage	0.8 V max.	0.8 V max.	0.8 V max.	0.8 V max.
Low Input Current	–0.4 mA max.	0.4 mA max.	0.4 mA max.	–0.4 mA max.
Clock Inputs				
High Input Voltage	2.4 V min.	2.4 V min.	2.4 V min.	2.0 V
Low Input Voltage	0.8 V max.	0.8 V max.	0.8 V max.	0.8 V
Min. Pulse Width	200 ns	600 ns (high) 600 ns (low)	600ns (high) 600ns (low)	25 ns
Max. Frequency	6 MHz	750 kHz	750 kHz	9 MHz
Gate Inputs				
High Input Voltage	2.4 V min.	2.4 V min.	2.4 V min.	2.4 V min.
Low Input Voltage	0.8 V max.	0.8 V max.	0.8 V max.	0.8 V max.
Outputs				
Fan Out	12 mA	12 mA	12 mA	12 mA
High Voltage Output	2.8 V min.	3.0 V min.	3.0 V min.	2.0 V min.
Low Voltage Output	0.6 V max.	0.4 V max.	0.4 V max.	0.8 V max.
High Output Current (Source)	2 mA	1 mA	1 mA	–12 mA max.
Low Output Current (Sink)	12 mA	2 mA	2 mA	12 mA max.

[1] Has same logic high and low voltage and current specifications as the digital I/O lines.

POWER, PHYSICAL, AND ENVIRONMENTAL SPECIFICATIONS

	KUSB-3100	KUSB-3102A	KUSB-3108	KUSB-3116	KUSB-3160
Power	<100 mA			±5%, @2 A max.	
+5V Enumeration Operation	<250 mA	100 mA max.	100 mA max.		100 mA max.
+5V Standby		0.5 mA max.	0.5 μA max.		0.5 mA max.
+5V Power On		500 mA max.	500 mA max.		500 mA max.
+5V Isolated Power Out (TB 27)		10 mA max.	10 mA max.		
Physical					
Dimensions	100mm (L) × 100mm (W) × 25mm (H)	6.5 in (L) × 4.5 in (W) × 1.4 in (H)	6.5 in (L) × 4.5 in (W) × 1.4 in (H)	190mm × 100mm	150mm × 100mm
Weight	4.8 oz. (136 g)	9 oz. (255 g)	9 oz. (255 g)	2 lbs. (906 g)	9 oz. (255 g)
Environmental					
Operating Temperature	0° to 70°C	0° to 55°C	0° to 55°C	0° to 55°C	0° to 50°C
Storage Temperature	–40° to 125°C	–25° to 85°C	–25° to 85°C	–25° to 85°C	–25° to 85°C
Relative Humidity	To 95% non-condensing	To 95% non-condensing	To 95% non-condensing	To 95% non-condensing	To 95% non-condensing
Certification and Compliance	FCC Part 15 Class B verified; will not compromise FCC compliance of host computer CE	FCC Part 15 Class B verified; will not compromise FCC compliance of host computer CE	FCC Part 15 Class B verified; will not compromise FCC compliance of host computer CE	FCC Part 15 Class B verified; will not compromise FCC compliance of host computer CE	FCC Part 15 Class B verified; will not compromise FCC compliance of host computer CE

1.888.KEITHLEY (U.S. only)

www.keithley.com

KEITHLEY

A GREATER MEASURE OF CONFIDENCE

ANEXO B. Listado de las funciones del software

El software de la interfaz gráfica está compuesto por dieciocho funciones **.m* que se encuentran adjuntas al presente Proyecto de Fin de Carrera en la carpeta donde se ha diseñado el software.

- Allchannels.m / Allchannels.fig
- askfrecuencia.m / askfrecuencia.fig
- Btool.m / Btool.fig
- Captura_manual.m / Captura_manual.fig
- Cargar.m
- Ciclos.m / Ciclos.fig
- Configuracion.m / Configuracion.fig
- Data_Pro.m / Data_Pro.fig
- fourier.m / fourier.fig
- Nivelesenergia.m / Nivelesenergia.fig
- parametros.m
- posicion_ejes.m
- registros.m
- registros_ciclos.m
- representacion.m
- Tarjeta_adquisicion.m / Tarjeta_adquisicion.fig
- tarjeta_conectada.m
- transformada_hilbert.m / transformada_hilbert.fig