UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Departamento de Ciencia e Ingeniería de Sistemas y Automática

INGENIERÍA TÉCNICA INDUSTRIAL: ELECTRÓNICA INDUSTRIAL

PROYECTO FIN DE CARRERA

DISEÑO DE UNA HERRAMIENTA CON MATLAB PARA LA ADQUISICIÓN Y PROCESAMIENTO DE SEÑALES. APLICACIÓN A SISTEMA DE DETECCIÓN DE FALLOS DE RODAMIENTOS.

Autor: Álvaro Sanz Arranz

Tutores: Ramón Ignacio Barber Castaño

Cristina Castejón Sisamón

LEGANÉS, OCTUBRE 2010

El sufrimiento es el precedente de la hazaña.

Álvaro Sanz

AGRADECIMIENTOS

En primer lugar quisiera agradecer muy especialmente a Ramón y Cristina, mis tutores del proyecto, la atención y apoyo que en todo momento me han brindado para poder llevar a buen puerto la tarea que en las próximas páginas se expone. Su paciencia y buenos consejos han sido determinantes a la hora de conseguirlo.

No puedo olvidar a todos los compañeros que han compartido conmigo el camino que hoy finaliza, tanto los que consiguieron su objetivo como los que se quedaron por el camino, cada uno de ellos y de ellas me han aportado unas enseñanzas que no se impartían en las horas docentes, y que no son otras que calidad humana, compañerismo y apoyo en los momentos menos buenos.

El máximo exponente de éste aspecto habéis sido vosotros dos, Alberto y Carlos; sin vuestra incondicional ayuda esto hubiera resultado mucho más duro de lo que ha sido. Comenzamos como compañeros y hemos finalizado como grandes amigos. Gracias.

Pero ante todo y sobre todo me tengo que rendir ante mi familia; mis padres, Antonio y Sagrario y mi hermano Javier.

Completamente seguro estoy de que sin vosotros no habría llegado siquiera al comienzo del estudio de esta titulación. Día a día habéis sido un soporte en el cual apoyarme, convenciéndome de que debía continuar en los innumerables momentos de flaqueza y ganas de abandonar.

Sois conscientes de que mis comienzos en la andadura que finaliza hoy fueron especialmente dubitativos, hasta puntos insospechados, pero vosotros enderezasteis mi senda, me hicisteis creer en mí mismo y me impulsasteis para poder ir venciendo uno a uno todos los obstáculos que, o bien me ponía yo mismo, o bien aparecían en el transcurso del tiempo.

Sois básicos en mi vida y quiero que seáis conscientes una vez más y que quede por siempre plasmado en estas páginas tan importantes para el futuro de mi existencia.

GRACIAS A TODOS!!!

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1: INTRODUCCIÓN

1.- INTRODUCCIÓN

1.1.- Motivación

Las motivaciones que este este proyecto ofrecía eran, desde un principio, variadas. La principal y más relevante hablaba de 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, pudiendo con dicha herramienta detectarlas y trabajar con vista a corregirlas, aumentando de este modo el confort y la durabilidad de los mismos.

Otro factor determinante que originó el presente proyecto es el que otorgaba la posibilidad de poder trabajar y desarrollar desde la base el diseño de una interfaz gráfica amigable y sencilla de utilizar por parte del usuario, que integrara la fase de adquisición de las señales con la de su procesamiento, en un área de trabajo que actualmente tiene gran relevancia, y que tiene trazas de continuar su línea ascendente en un futuro no excesivamente lejano.

La colaboración y contacto con el departamento de Ingeniería Mecánica fue un factor que me motivó personalmente, ya que en mi familia hay antecedentes que trabajan dentro de ésta disciplina y es algo que quería tener presente a la hora de elaborar mi proyecto fin de carrera.

1.2.- Objetivos

Este proyecto plantea sin duda una serie de objetivos realmente ambiciosos desde el punto de vista de la aplicación práctica que va a suponer en un futuro. Han colaborado los departamentos de Mecánica y de Automática y Sistemas de la Escuela Politécnica Superior de la Universidad Carlos III de Madrid para así poder desarrollar y diseñar una interface de usuario gráfica que permita dinamizar y ampliar el estudio de señales procedentes de distintos sensores, con el fin de analizarlas en la misma interface. La aplicación para la que se concibió la herramienta fue para el análisis de señales procedentes de acelerómetros con el fin de detectar fallos en rodamientos.

El objetivo principal de este proyecto es la realización de una interface gráfica en Matlab que permita las fases de captura de señales y procesamiento de señales destinadas a la detección de fallos en rodamientos. Este objetivo se particulariza en:

Los objetivos propuestos para este proyecto son:

- Diseño de una interface gráfica que permita la adquisición de datos utilizando la tarjeta KUSB 300 de Keithley.
- Diseño de una interface de usuario conjunta que integre las lecturas de las señales con los algoritmos.

- Integración de los primeros algoritmos de detección de fallos.

La herramienta de ha sido concebida de forma amplia para que permita el estudio de señales tomadas desde una MFS Lite para medir vibraciones mecánicas en rodamientos (que es la principal aplicación que se le dará en el futuro y la que movió las principales directrices a la hora de programar la herramienta) hasta señales eléctricas y electrónicas en general (fuentes de tensión AC/DC, transitorios, evoluciones temporales de cargas eléctricas...etc.).

Las señales expuestas son sólo ejemplos, ya que, con la sola ayuda de un PC y una tarjeta de adquisición de datos, será posible realizar mediciones de cualquier magnitud, previa conversión a estímulos electrónicos, medible, pudiendo representarla en función del tiempo para posteriormente poder tratarla y llevar a cabo diversas transformaciones de la misma con el objeto de realizar un estudio completo de las evoluciones y características particulares de las mismas.

Además, se buscaba que se pudieran medir varias señales simultáneamente y en tiempo real. El objetivo era, por tanto, incrementar las prestaciones del software desarrollado, al permitir llevar a cabo el estudio de un rango comprendido entre uno y ocho canales, a elegir por el usuario del mismo.

Se adjuntan además en el proyecto directrices acerca de la instrumentación necesaria para una buena toma de datos, y una extensa descripción de la importancia de los objetivos que se podrán conseguir gracias al empleo de la interfaz gráfica desarrollada, con gran sencillez por parte del usuario

1.3.- Partes del documento

Como primera toma de contacto con el trabajo realizado se ha realizado el estudio del mantenimiento predictivo, explicándolo para que el lector, sobre todo el ajeno al área de Ingeniería Mecánica, pueda constatar por sí mismo la importancia de la disciplina, pudiendo de este modo comprender mejor que el proyecto gire en torno a ella, y que el diseño y las prestaciones de la aplicación desarrollada vayan siempre buscando satisfacer de la mejor forma posible el estudio de una parte importante del mantenimiento predictivo, como es el tratamiento de vibraciones mecánicas en rodamientos.

A continuación se pasa a detallar formalmente el procesamiento de señales, clasificando los tipos presentes en el estudio de las mismas, el modo de procesarlas y finalmente los modos y herramientas de los que se dispone actualmente para llevar dicha tarea a buen puerto. Entre los métodos más notables, y por ello mismo, en los que se centra el trabajo realizado se encuentran la transformada de Hilbert, el espectro de frecuencias y la transformada Wavelet.

Se continúa concretando la forma en la que se va a llevar a cabo el estudio de las señales vibratorias, describiendo los elementos que entran en juego para tal fin, previa introducción en conceptos importantes como son los tipos de rodamientos y las causas que suelen originar el deterioro prematuro de los mismos.

Acto seguido se abre un apartado en el que la programación de interfaces gráficas de usuario mediante la herramienta GUI de MATLAB es el tema predominante. Se indica pormenorizadamente en lo que consiste una interfaz gráfica, los tipos de archivos con sus respectivas extensiones que aparecen en el desarrollo de las mismas, el tratamiento que ha de darse a los mismos y, por último, nociones acerca del uso de la herramienta GUI, así como un ejemplo de programación sencillo para una mejor asimilación de los conceptos expuestos.

Una vez conocido tanto el entorno de trabajo como los objetivos que se buscan con la realización del mismo, se describirá la aplicación que ha sido desarrollada como fruto del trabajo que se expone en estas páginas. Se muestran todas las aplicaciones y utilidades que ofrece la herramienta Btool, y se llevan a cabo demostraciones con el programa trabajando en modo simulación, es decir, sin tener conectado aún el sistema.

Se pasa a continuación a mostrar 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 de la misma.

Por último se dan 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.

CAPÍTULO 2: ESTADO DEL ARTE

2.- ESTADO DEL ARTE

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 [1]:

- *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 [2].

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.

Y 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, lo que evita detener la máquina debido a una avería y su consiguiente reparación, lo que sin duda supone un importante ahorro de la pérdida de producción.

Para determinar la condición de la máquina se evalúa una serie de síntomas que la misma emite al exterior, es por ello por lo que 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, ultrasonidos... Entre otras.

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 [2].

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.

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 pues, 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 ello por lo 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 [1].

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.

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 [2].

- *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 [2]:

- 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 (figura 1). Los defectos originados por vibraciones pueden clasificarse como localizados y distribuidos [1]:

- *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.

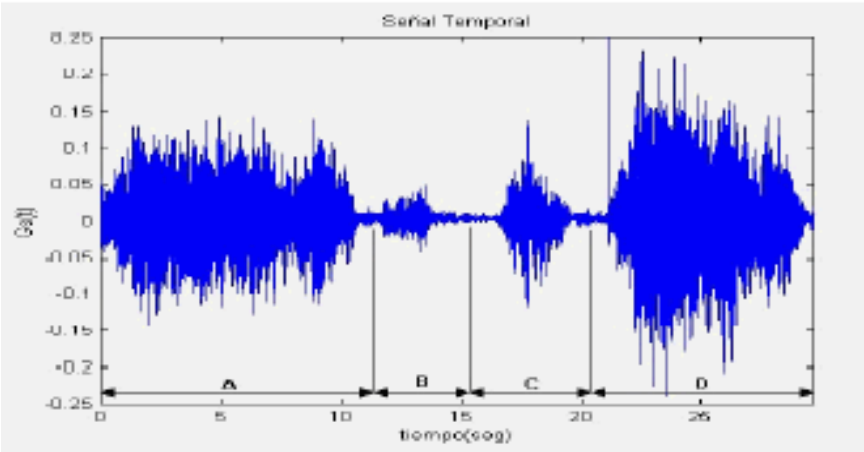


Figura.1: Registro de las vibraciones en un ciclo de trabajo en función del tiempo

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 [2]:

- *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 imposibole 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 ele studio 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 (figura 2), provenientes de filtros y sellos defectuosos o el uso de un producto lubricante inadecuado atendiendo a las especificaciones de la máquina.

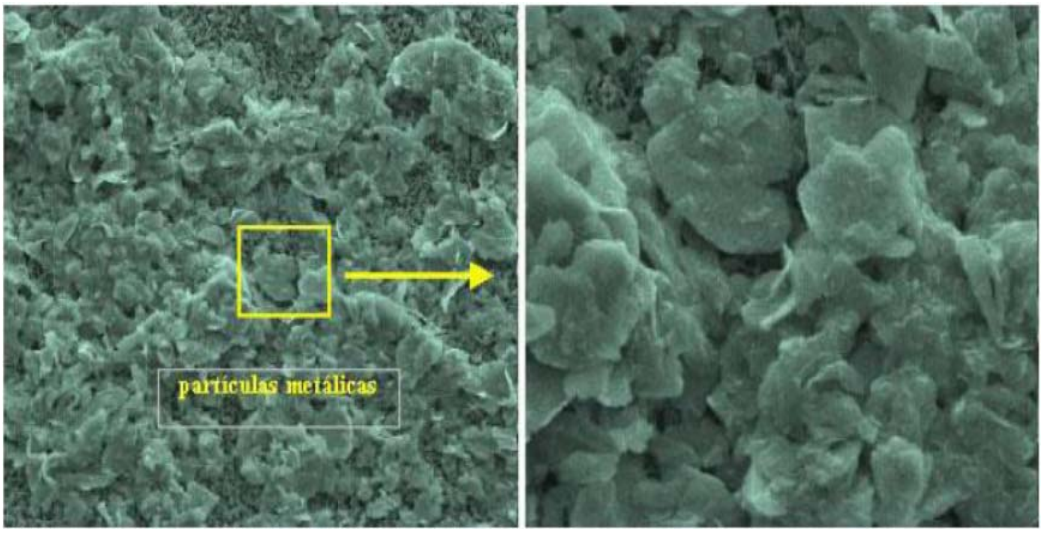


Figura 2: Presencia de partículas sólidas en el análisis de aceites

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*), ver 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.

Además, 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 [5]:

- 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 [3].

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.

En la 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.

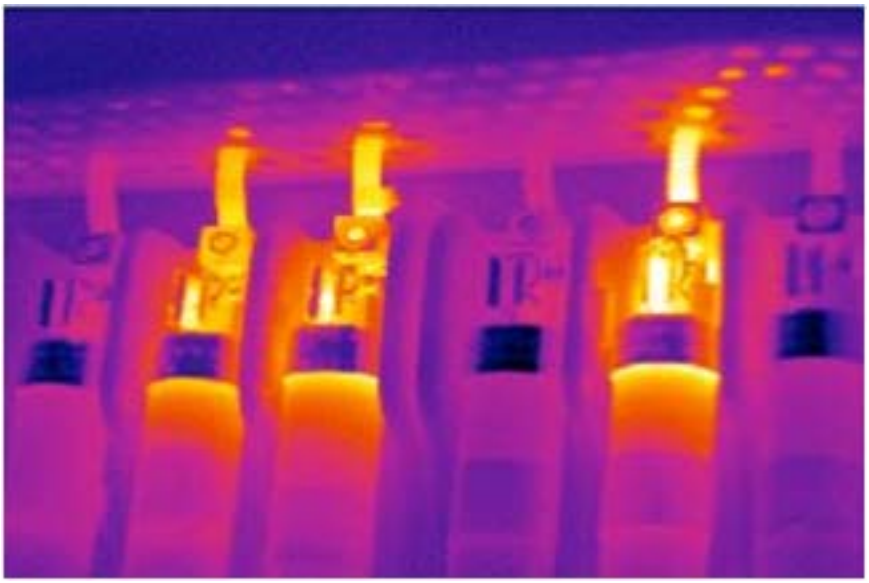


Figura.4: Imagen obtenida por Termografía de un cuadro de mando

Análisis por corriente eléctrica

El objetivo del análisis eléctrico como técnica de mantenimiento predictivo es el de realizar estudios eléctricos (figura 5) 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 [2].

En función de la corriente de alimentación, trifásica o continua, del equipo (generalmente se trata de motores eléctricos) 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.

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 [4].

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* [11].

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 figura6, las señales en general se dividen en dos grandes grupos [11]:

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 que se presenta en la figura 6.

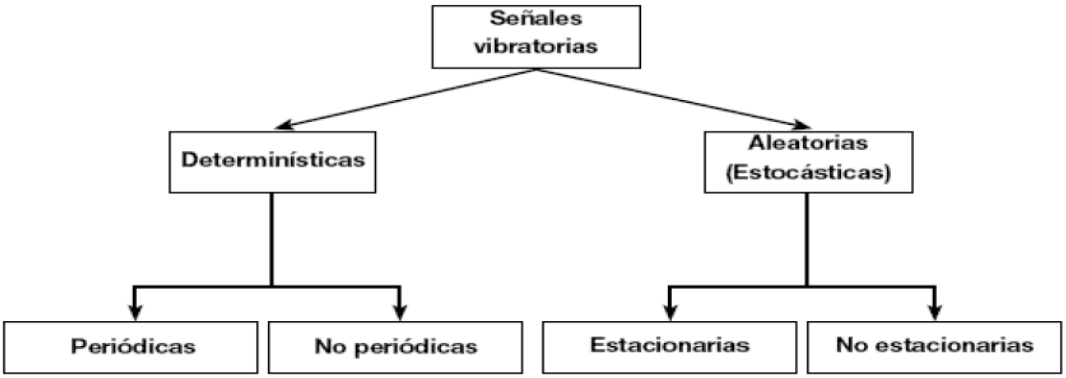


Figura 6: Clasificación de las señales vibratorias

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 en la figura 7 [8].

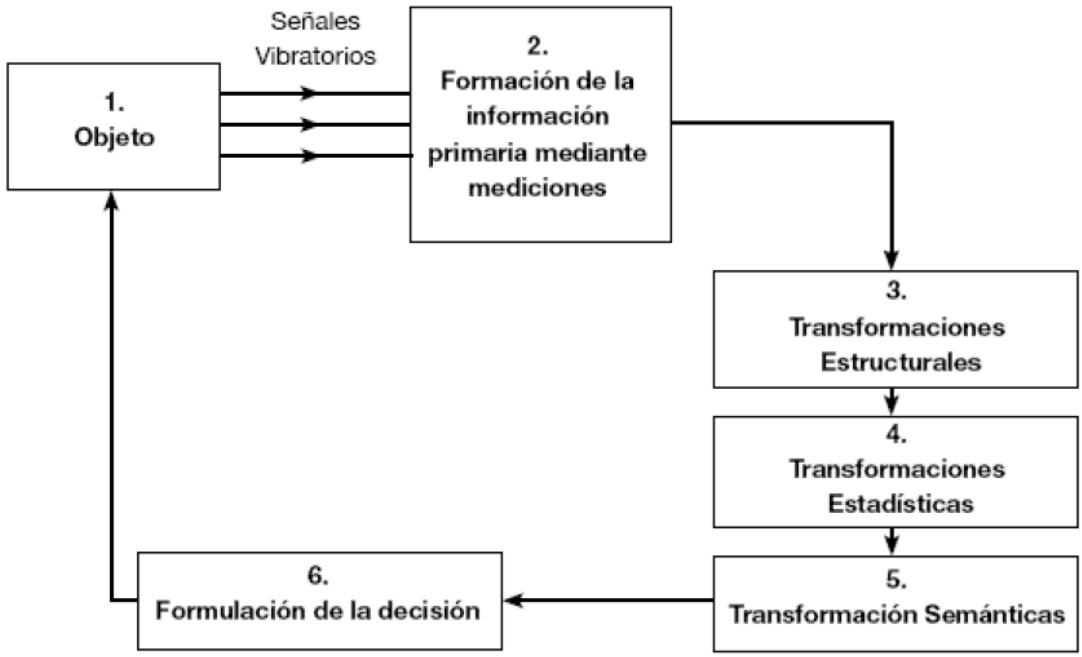


Figura.7: Metodología para el procesamiento de señales vibratorias

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 a fines de los años 60 (figura 8), aunque todavía hay empresas que los siguen utilizando [12].

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

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 (figura 9) 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.

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

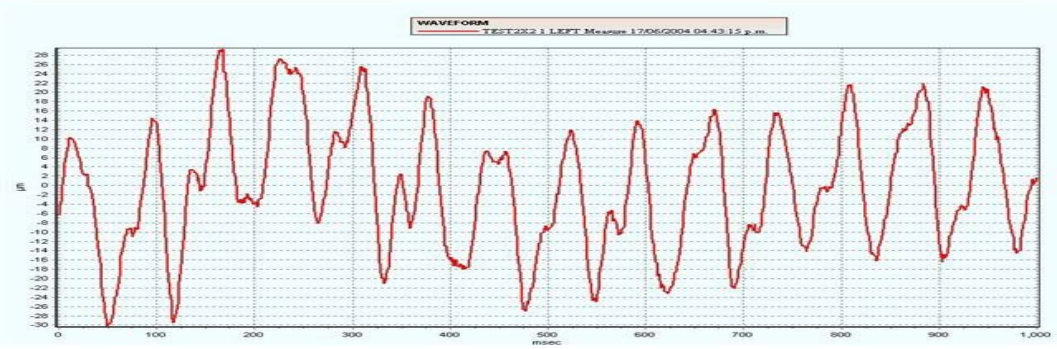


Figura 9: Análisis de la forma de onda de una vibración

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, ver figura 10.

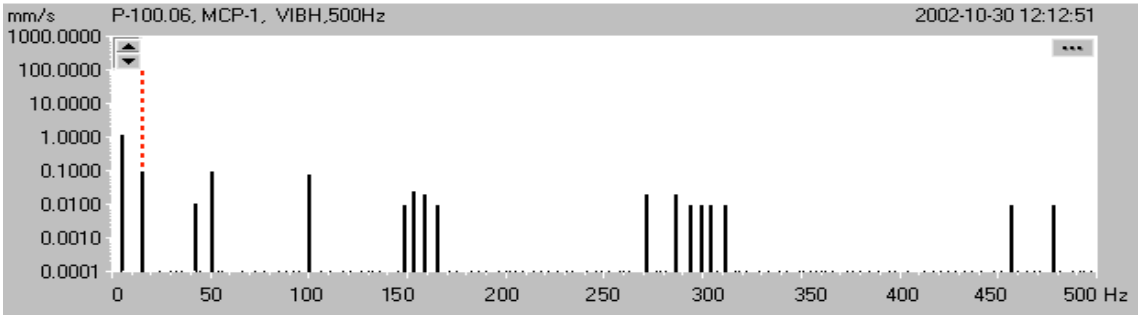


Figura 10: Vibraciones a distintas frecuencias

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" [12] (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), ver figura 11.

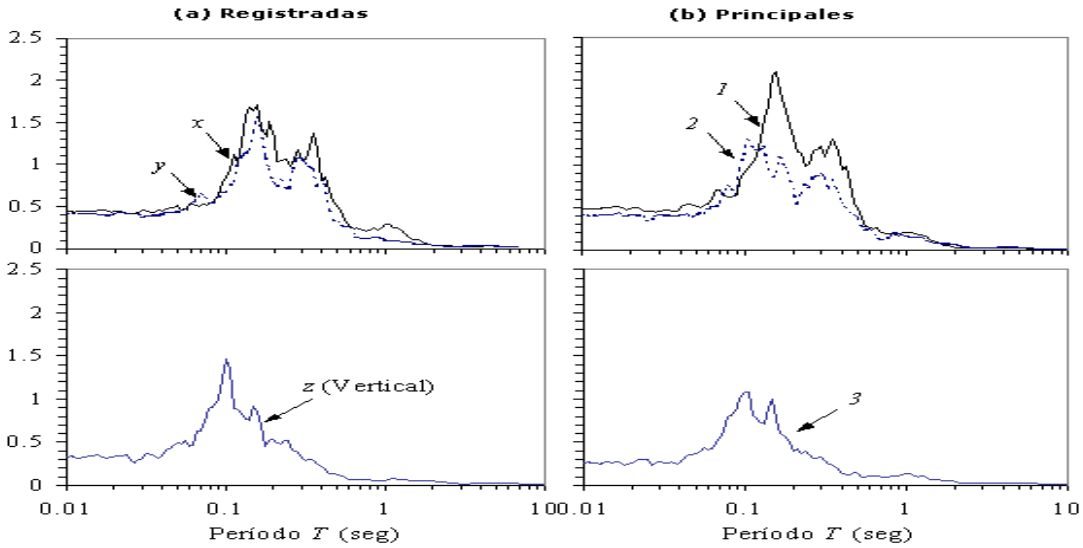


Figura 11: Variación temporal de las vibraciones

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 (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).

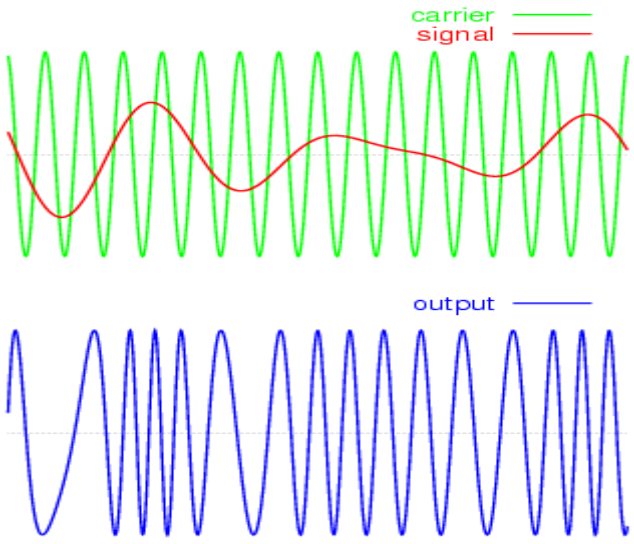


Figura 12: Modulación FM de una señal mostrando las señales portadora y modulada

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, ver 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 [12].

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*. [7] 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 remuestreo 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 espectrograma

La *transformada de Fourier (FT)* es una herramienta matemática que sirve para describir el comportamiento de señales no periódicas [13].

Su objetivo es la descomposición de la función en una suma de funciones armónicas.

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

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

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

De manera simbólica se puede decir que:

$$x(t) \leftrightarrow X(f)$$

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 [13]. 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 [11]:

"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.- Espectrograma

El módulo de la transformada corta de Fourier (STFT) vista en el apartado anterior elevado al cuadrado se denomina espectrograma [12] 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.

Figura 14: Espectrograma en dos dimensiones

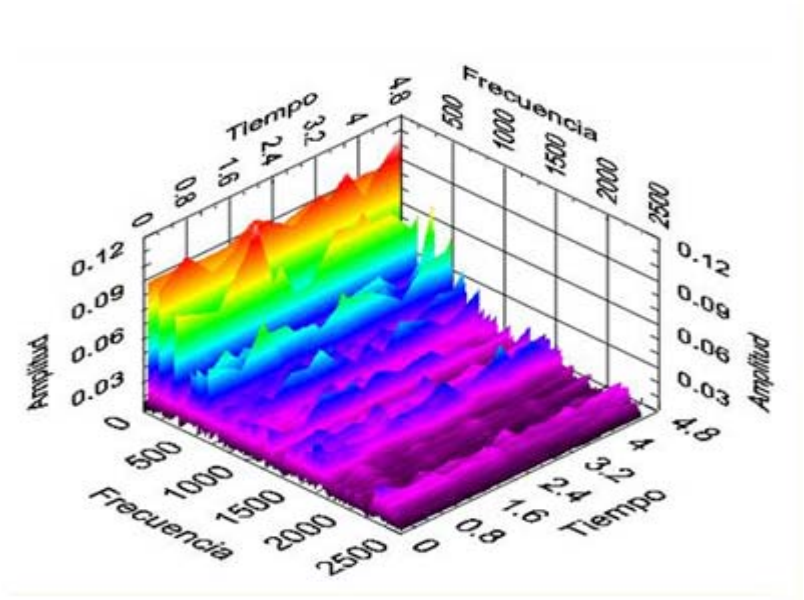


Figura 15: Espectrograma en tres dimensiones

En las figuras 14 y 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 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 [13].

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 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* [14], ya que proporcionará una resolución variable en el tiempo.

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 multirresolución, 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 [15], 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... entre otros.

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 [15]:

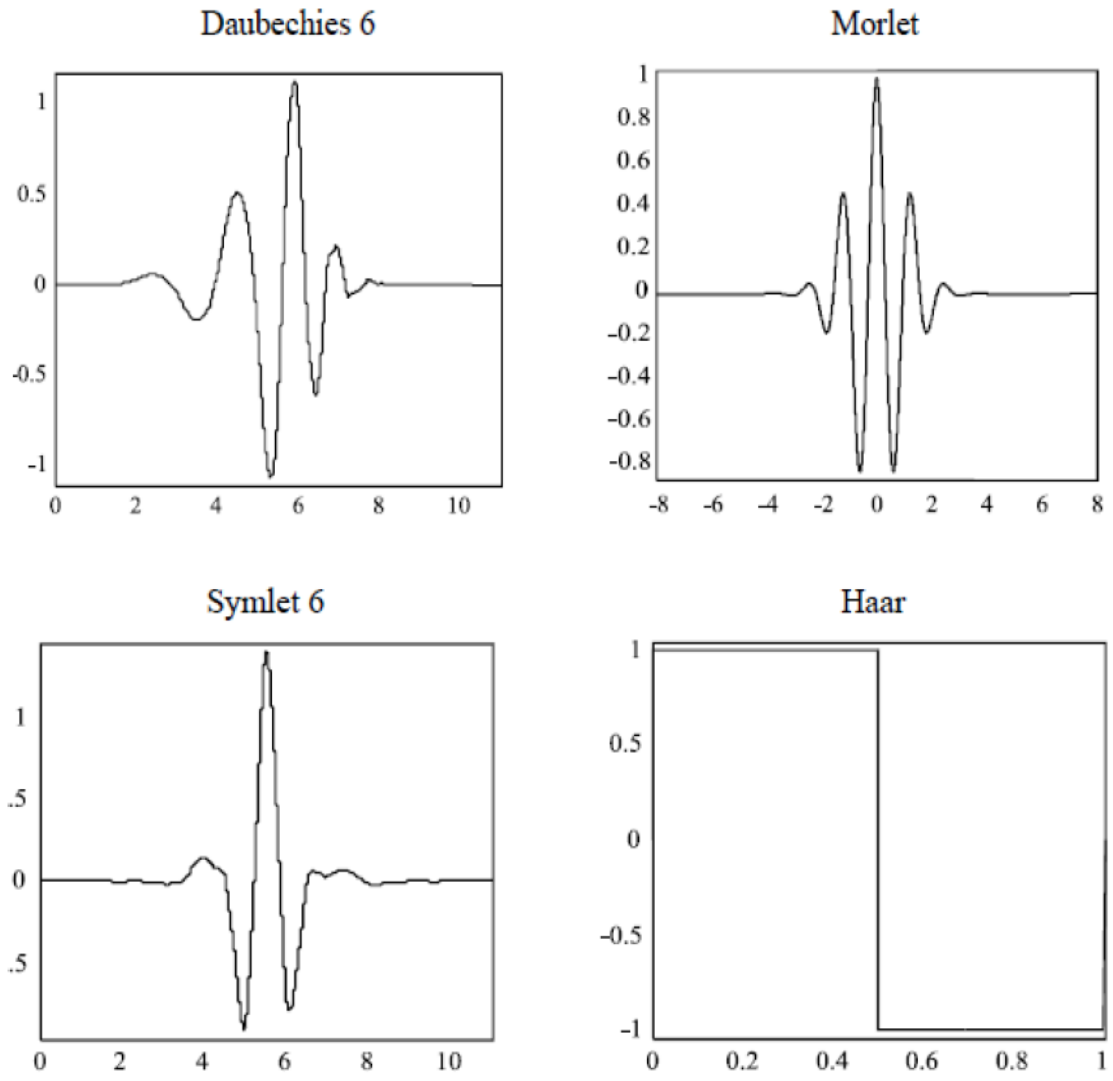
$$h_{a,b} = \frac{h\left(\frac{x-b}{a}\right)}{\sqrt{|a|}}; a,b \in R, a \neq 0$$

Son generadas a partir de funciones madre *h(x) (Figura).* 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 [14] se muestran en la figura 16.



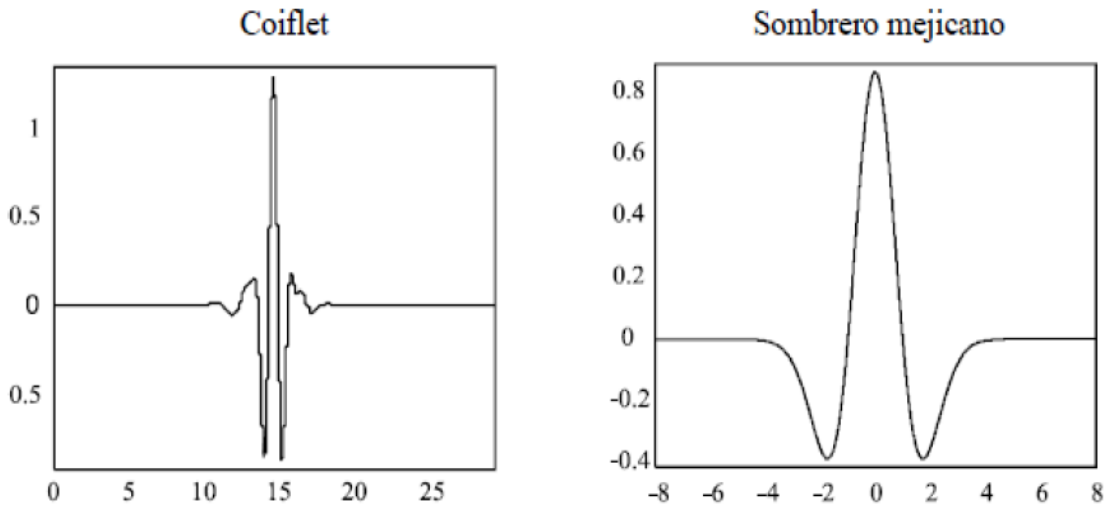


Figura 16: Wavelets madre más usuales

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 [15].

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 [14]. Se define como:

$$d_{j,k} = a_0^{\frac{j}{2}} \int f(x) h(a_0^{-j} x - k b_0) dx$$

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.

- El *decimado* se refiere a incrementar el número de muestras que se toman.
- El *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 $d_{j,k}$.

La energía de los coeficientes de escala c_k 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:

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

CAPÍTULO 3: SISTEMA DE DETECCIÓN Y DIAGNOSIS DE VIBRACIONES

3.- SISTEMA DE DETECCIÓN Y DIAGNOSIS DE VIBRACIONES

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.

Seguidamente se pasará a mostrar el equipo necesario para la realización del ensayo, describiendo brevemente cada uno de los elementos que entran en juego y los aspectos técnicos más relevantes de cada uno de ellos.

Seguidamente se describirá la forma en la que se van a capturar los datos, es decir, la frecuencia de muestreo empleada, las velocidades de rotación del eje a la que se van a recoger los datos y la cantidad de datos que necesita cada velocidad para tener una muestra representativa.

3.1.- Los 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 [12].

Estos cuerpos rodantes suelen ir sujetos en la jaula. En la *Figura 17* se muestran los principales elementos que componen un rodamiento.

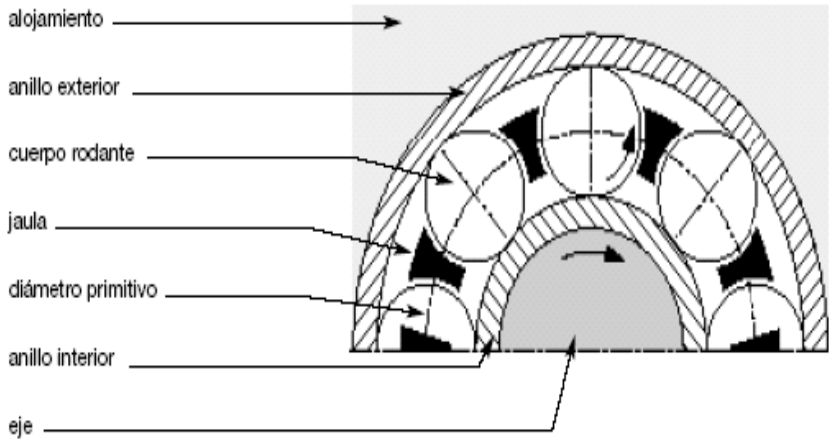


Figura 17: Elementos que componen un rodamiento

3.1.2.- Tipos de rodamientos

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

Figura 18: Tipos de rodamientos

3.1.3.- Tipos de defectos y causas

Los tipos más significativos de fallos que se pueden dar en rodamientos son [13]:

- *Desgaste*: La presencia de partículas abrasivas (pequeñas identaciones 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, tal y como se observa en la *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, tal y como puede observarse en la *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. Se muestra en la *Figura 21.*

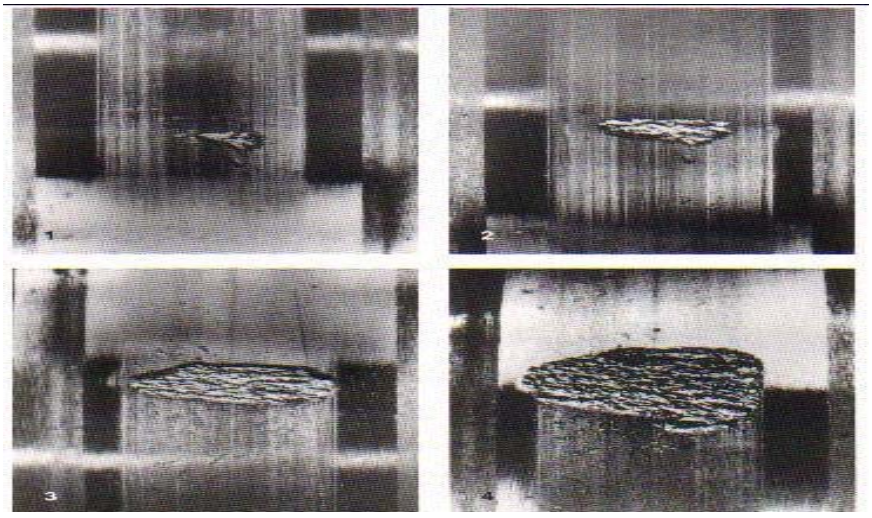


Figura 21: Descascarillado en rodamientos

- *Smearing:* Se produce cuando dos superficies lubricadas incorrectamente deslizan una sobre otra en condiciones de elevada carga. En este caso se originan microsoldaduras 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, tal y como se aprecia en la *Figura 22*.

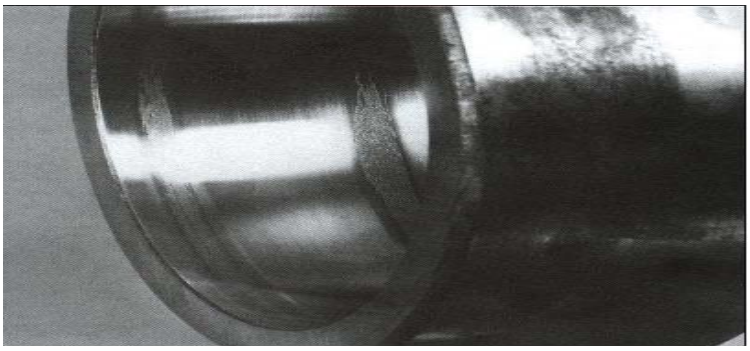


Figura 22: Smearing en un rodamiento

- *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. *(Figura 23)*

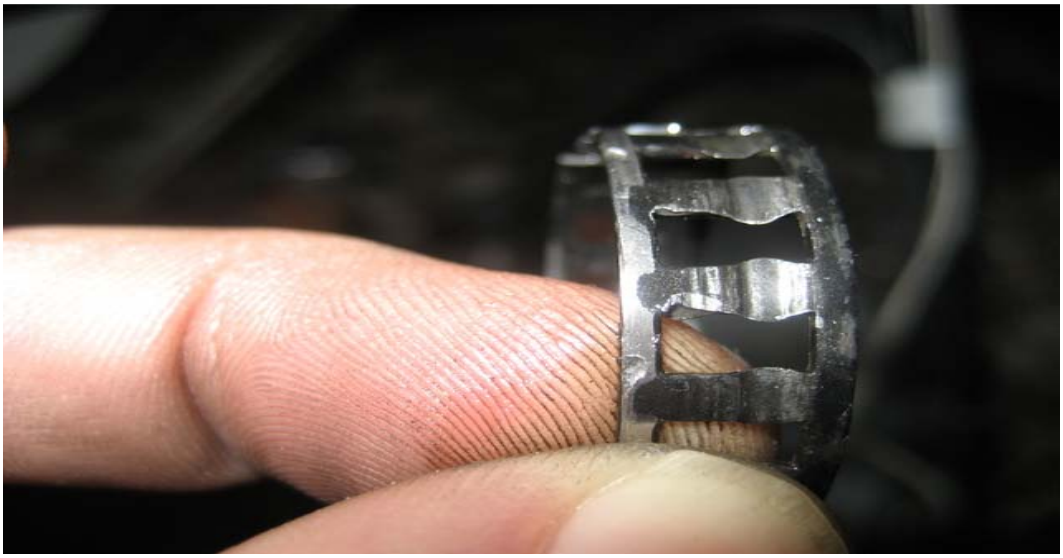


Figura 23: Rodamiento con la jaula deformada

En la *Figura 24* se adjunta la representación de los porcentajes pertenecientes a cada una de las causas de los defectos en rodamientos [13].

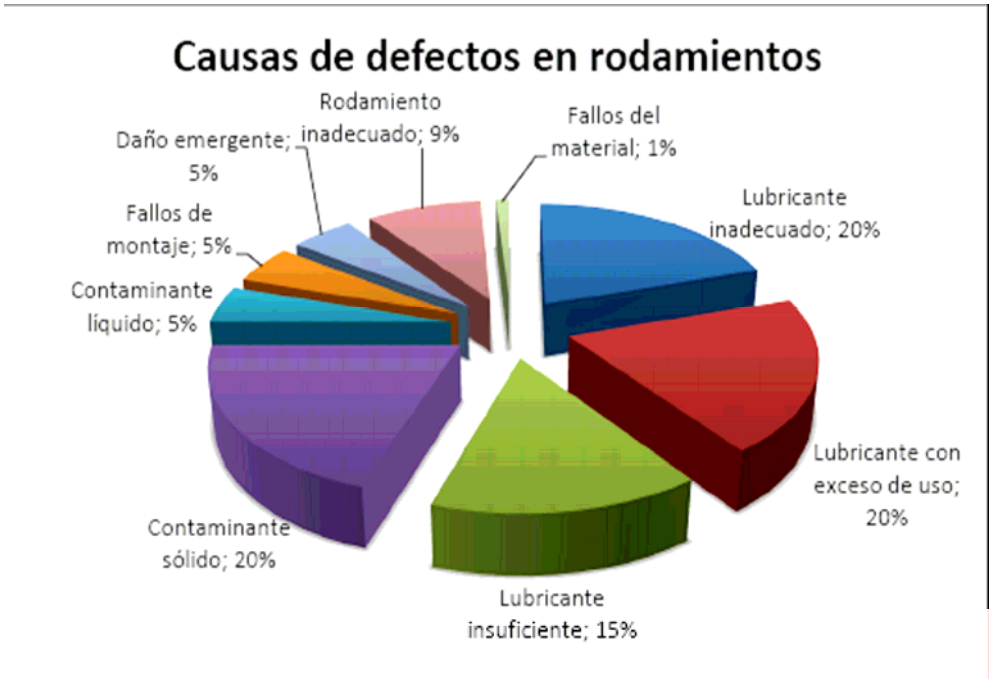


Figura 24: Causas de defectos en rodamientos y porcentajes

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.

Otra causa a tener en cuenta será la presencia de sustancias o partículas contaminantes en el ámbito de trabajo.

El resto de razones no hay que dejarlas de observar, pero no dejan de ser anecdóticas frente a las expuestas anteriormente.

3.2. El sistema de ensayos: Machine Fault simulator Lite (MFS Lite)

Máquina de ensayo de rodamientos (*Figura 25*). 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.

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.

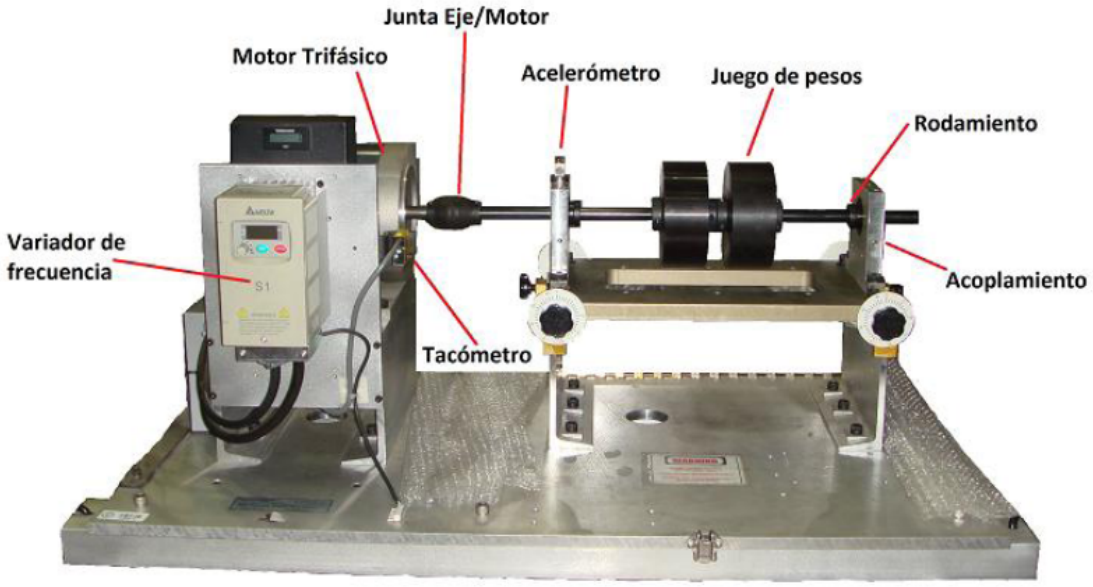


Figura 25: MFS Lite

3.3. 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.

Puede experimentar tres tipos de deformación mecánica, que son:

- 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 como el que se muestra en la *Figura 26*.

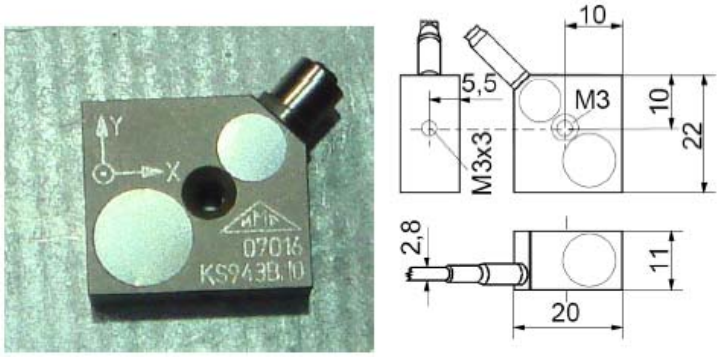


Figura 26: Acelerómetro triaxial modelo KS-943B.10

El acelerómetro empleado ocupará tres canales de la aplicación *Btool*, que podrán ser representados en tiempo real y que 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.

3.4. Tarjeta de adquisición de datos

La tarjeta de adquisición de datos es de la marca Keithley modelo KUSB 300. Su función es la de convertir la señal analógica adquirida por los sensores en una señal digital para su posterior procesado en un ordenador.*(Figura 27)*

Figura 27: Tarjeta de adquisición de datos

3.5. Filtros y amplificadores

El propósito de un filtro es remover señales indeseadas desde la señal que se está trabajando *(Figura 28).*

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 antialiasing.

Igual que un filtro de ruido, el filtro antialiasing 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-

Figura 28: Filtro-amplificador

3.6 Tacómetro

Junto con un elemento reflectante dispuesto en el eje del rotor, permite registrar la velocidad de giro del motor. Se muestra un tacómetro digital en la *Figura 29.*

Figura 29: Tacómetro digital

3.7 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, dirigirse al *Apartado 5*,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 rodamientos.

CAPÍTULO 4: INTERFACES DE USUARIO CON MATLAB

4. INTERFACES DE USUARIO CON MATLAB

4.1 MATLAB

MATLAB © (abreviatura de *MATrix LABoratory*, "laboratorio de matrices"), figura 30, es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M) [9]. Está disponible para las plataformas Unix, Windows y Apple Mac OS X.

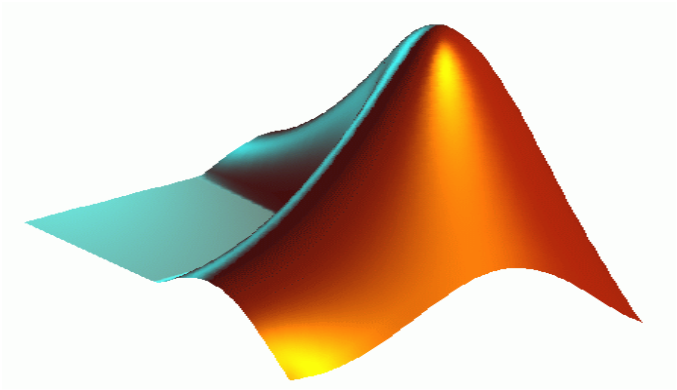


Figura 30: Logotipo de MATLAB

Fue creado por *Cleve Moler* en 1984, surgiendo la primera versión con la idea de emplear paquetes de subrutinas escritas en Fortran en los cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje.

El lenguaje de programación M fue creado en 1970 para proporcionar un sencillo acceso al software de matrices *LINPACK* y *EISPACK* sin tener que usar Fortran.

En 2004, se estimaba que MATLAB era empleado por más de un millón de personas en ámbitos académicos y empresariales.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.

El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, que son Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI).

Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (*toolboxes*); y las de Simulink con los paquetes de bloques (*blocksets*).

MATLAB es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

Las funcionalidades de Matlab se agrupan en más de 35 cajas de herramientas y paquetes de bloques *(para Simulink)* [9], clasificadas en las categorías mostradas en la *Figura 31*.

MATLAB (Cajas de herramientas)	Simulink
Matemáticas y optimización	Modelado de punto fijo
Estadística y análisis de datos	Modelado basado en eventos
Diseño de sistemas de control y análisis	Modelado físico
Procesado de señal y comunicaciones	Gráficos de simulación
Procesado de imagen	Diseño de sistemas de control y análisis
Pruebas y medidas	Procesado de señal y comunicaciones
Biología computacional	Generación de código
Modelado y análisis financiero	Prototipos de control rápido y SW/HW HIL
Desarrollo de aplicaciones	Tarjetas integradas
Informes y conexión a bases de datos	Verificación, validación y comprobación

Figura 31: Funcionalidades de MATLAB

4.2. Guides de MATLAB

El presente proyecto se centra en una de las utilidades más valiosas del paquete MATLAB, como es la creación de interfaces de usuario con un gran abanico de posibilidades y gran sencillez para el usuario de las mismas una vez diseñadas mediante MATLAB GUIDE.

GUIDE es un entorno de programación visual disponible en MATLAB para realizar y ejecutar programas que necesiten ingreso continuo de datos.

Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++, pero haciendo más intuitiva e inmediata la programación y el acceso a las diferentes funcionalidades que ofrece.

La interfaz gráfica de usuario, conocida también como GUI (del inglés *graphical user interface*) es un programa informático que actúa de canal de comunicación entre humano y ordenador, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la interacción con el sistema operativo de una máquina o con un ordenador sin la necesidad de poseer grandes conocimientos al respecto.

Se busca obtener la máxima simplicidad posible en la comunicación, ya que ésto repercutirá en una ampliación directamente proporcional del rango de usuarios que podrán tener acceso a las aplicaciones.

Habitualmente las acciones se realizan mediante manipulación directa [9]. Surge como evolución de los intérpretes de comandos que se usaban para operar los primeros sistemas operativos y es pieza fundamental en un entorno gráfico.

Como ejemplos de interfaz gráfica de usuario, cabe citar los entornos de escritorio Windows, el X-Window de GNU/Linux o el de Mac OS X, conocido como Aqua.

En el contexto del proceso de interacción persona-ordenador, la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

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.

Atendiendo a como el usuario puede interactuar con una interfaz, se dan varios tipos de interfaces de usuario:

- *Interfaces alfanuméricas (intérpretes de mandatos)* que solo presentan texto.

- *Interfaces gráficas de usuario (GUI),* las que permiten comunicarse con el ordenador de una forma muy rápida e intuitiva representando gráficamente los elementos de control y medida.

- *Interfaces táctiles*, que representan gráficamente un "panel de control" en una pantalla sensible que permite interaccionar con el dedo de forma similar a si se accionara un control físico.

El proyecto se centra en el diseño y desarrollo de una interfaz gráfica de usuario.

4.3.- Componentes de las guides

Se trata del espacio desde el cual seleccionar los componentes que se precisan de acuerdo con las diferentes funcionalidades que éstos presentan. Se puede acceder directamente, pues se detallan los iconos en la parte izquierda del entorno de diseño GUI *(Figura 4)* o mediante el menú *Layout editor*.

Los componentes disponibles se detallan a continuación:

- **Check box**

Indica el estado de una opción o atributo; puede estar activado o desactivado, y en función de ese estado se actuará de un modo u otro. Su valor de estilo es *checkbox.*

- **Editable text**

Se trata de un cuadro de diálogo por el cual introducir datos que posteriormente se tratarán mediante la lectura de los mismos por parte del programa. Su valor de estilo es *edit.*

- **Pop-up menu**

Se trata de un desplegable en el que se dan varias opciones a elegir en forma de lista, y del cual sólo se podrá seleccionar una opción. Su valor de estilo es *popupmenu.*

- **List box**

Se trata de una presentación de opciones igual que en el caso anterior, sólo que en este caso se presentan en forma de lista deslizable por medio del ratón. Su valor de estilo es *listbox.*

- **Push button**

Se trata de un botón que, al ser pulsado, invoca el evento al que está asociado inmediatamente. Su valor de estilo es *pushbutton.*

- **Radio button**

Se trata de otra presentación posible equivalente a check box, que permite pues seleccionar una determinada opción mediante la activación o desactivación de las mismas. Su valor de estilo es *radio.*

- **Toggle button**

Se trata de un botón con dos estados, que son on y off. Su valor de estilo es *togglebutton.*

- **Slider**

Se trata de una visualización intuitiva para representar valores mediante el desplazamiento de los mismos por medio del ratón. Su valor de estilo es *slider.*

- **Static text**

Se emplea para poner etiquetas mediante una cadena de caracteres, mostrando un string de texto en el interior de un recuadro. Su valor de estilo es *text.*

- **Panel button**

Se emplea para agrupar una serie de botones relacionados entre sí por medio de un contexto común en un sólo grupo.

- **Button group**

Permite exclusividad de selección con los radio button mediante el agrupamiento de los mismos.

4.4.- Programación de Guides

4.4.1.- Introducción

En éste apartado se van a exponer y explicar pormenorizadamente todos los aspectos relacionados con la programación con Guides de MATLAB, así como unos ejemplos que simplifiquen la asimilación de conceptos y unas directrices para comenzar a usar el programa de forma básica [9].

El primer factor que se debe de tener en cuenta es que la programación de divide en dos aspectos; uno más gráfico y otro de programación pura.

Una vez se haya llegado al punto en el que se ha alcanzado el aspecto de la interfaz gráfica de usuario que se deseaba y se han incorporado todas las utilidades que los requerimientos exigen para el cumplimiento de las premisas impuestas por el cliente, se generarán automáticamente dos ficheros, con las extensiones y características que se detallan a continuación:

- Fichero *.fig*:

  Es el fichero que comprende los aspectos visuales de la interfaz de usuario. En él se registran todos los componentes que conforman la aplicación, así como sus propiedades y algo muy importante, como es el *Tag* o *etiqueta* de cada elemento de la GUI, mediante el cual se podrá identificar el mismo para la posterior programación en el archivo .m.

  Se podrán modificar los aspectos de este fichero en cualquier momento mediante la apertura del editor del mismo, que se pasará a explicar más adelante.

- Fichero *.m*:

  Es un fichero en formato de texto que tiene una parte generada por el programa, con aspectos tales como las funciones de inicialización, los callbacks (se explicarán más adelante) y las funciones para cada elemento de la GUI, con el objetivo de que el usuario pueda programar cada uno de los elementos acorde con las necesidades que precise a cada momento, de modo que en función de lo añadido por el diseñador, el programa actúe de una determinada forma u otra.

Los dos ficheros se relacionan a través de las conocidas como subrutinas *callback*. En programación, un *callback* es un código ejecutable que es pasado como un argumento a otro código [16].

Permite que una capa de software de nivel inferior llame a una subrutina o función definida en una capa de nivel superior de un modo dinámico y realmente sencillo.

Un *callback* puede ser usado como una alternativa más simple al polimorfismo y a la programación genérica, en donde el comportamiento exacto de una función puede ser dinámicamente determinado pasando diferentes punteros de funciones a una función de nivel inferior. Esta puede ser una técnica poderosa de reutilización de código.

Una vez que se graba, los archivos desde la consola de emisión se puede ejecutar el programa resultante en la ventana de comando de *Matlab* solamente escribiendo el nombre del archivo.

Por ejemplo si se guarda un archivo de una GUI denominado como *ej.fig* y *ej.m* , respectivamente, escribiendo *ej* y presionando *enter* se ejecuta el programa.

4.4.2.- Generación del fichero . fig

A continuación se pasará a exponer el programa empleado en el desarrollo del presente proyecto en cuanto a su orientación para tratar y generar el archivo *.fig* (por añadiruría se tendrá también el archivo *.m* que se explicará en el siguiente apartado), así como unas básicas indicaciones acerca de su utilización [10].

Se accede al GUIDE a través de la previa ejecución de MATLAB, y se puede hacer de dos formas distintas:

- Mediante la escritura de la instrucción “guide” en la ventana de comandos.
- Haciendo clic en el icono correspondiente a GUIDE *(Figura 32).*

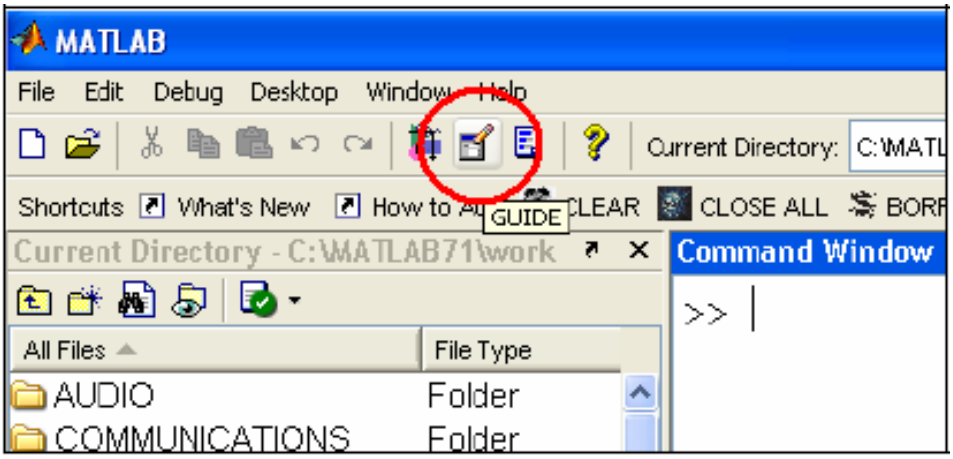


Figura 32: Icono GUIDE dentro de MATLAB

Una vez llevada a cabo una de las dos opciones se presentará el cuadro de diálogo representado en la *Figura 33.*

Dentro de dicho cuadro de diálogo se presentan las siguientes opciones [10]:

1) **Blank GUI (Default)**
Se trata de la opción de interfaz gráfica de usuario en blanco predeterminada, por ser la más utilizada. Presenta un formulario nuevo en el cual se podrá diseñar un determinado programa desde la base.

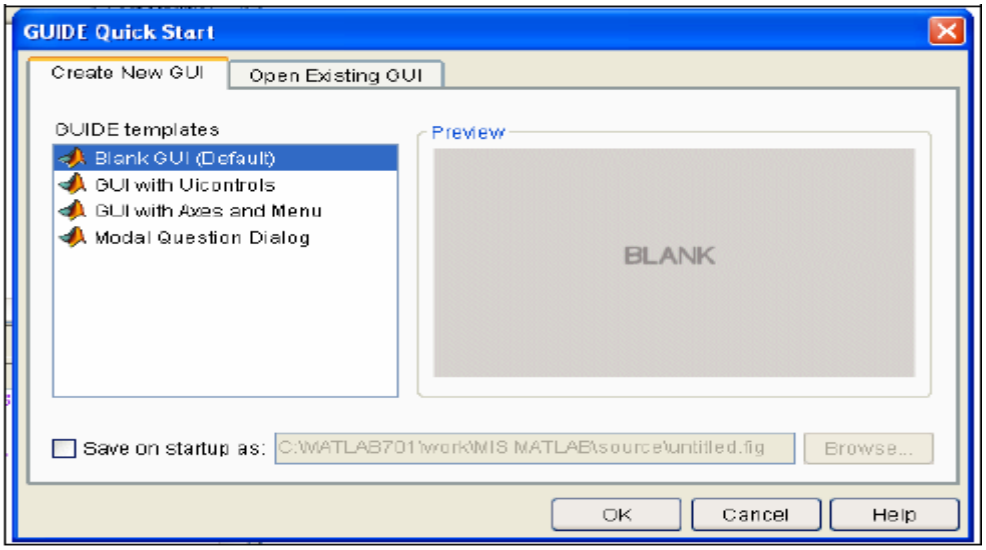


Figura 33: Ventana de inicio de GUI

2) GUI with Uicontrols

Esta opción presenta un ejemplo en el cual se calcula la masa, dada la densidad y el volumen del objeto, en cualquiera de los dos sistemas de unidades más extendidos. Seleccionando dicha opción se puede ejecutar el ejemplo y obtener resultados a partir del mismo.

3) GUI with Axes and Menu

Esta opción es otro ejemplo en el cual se incluye un menú *File* con las opciones *Open*, *Print* y *Close* en su interior. En el formulario cuenta con un *Popup menu*, un *Push button* y un objeto *Axes* (Se explicarán estos objetos más adelante).

Se puede ejecutar el programa eligiendo alguna de las seis opciones que se encuentran en el menú desplegable, para posteriormente hacer clic en el botón de comando.

4) Modal Question Dialog

Con esta opción se muestra por pantalla un cuadro de diálogo común, que consta de una pequeña imagen, una etiqueta y dos botones, con las opciones *Yes* y *No* respectivamente.

Dependiendo del botón que se presione, el GUI en cuestión devolverá el texto seleccionado (la cadena de caracteres *Yes* o *No*).

Por norma general, y salvo casos excepcionales, se escoge casi en la totalidad de las ocasiones la opción 1, pues lo más extendido es el diseño de interfaces gráficas que se adecuen a las necesidades del cliente, y no el uso de interfaces ya predeterminadas por MATLAB GUI.

Tras seleccionar la opción *Blank GUI (Default)* se tendrá, figura 34 el entorno de diseño presente al iniciar un GUI en blanco, pudiéndose diferenciar tres zonas que a continuación se detallan:

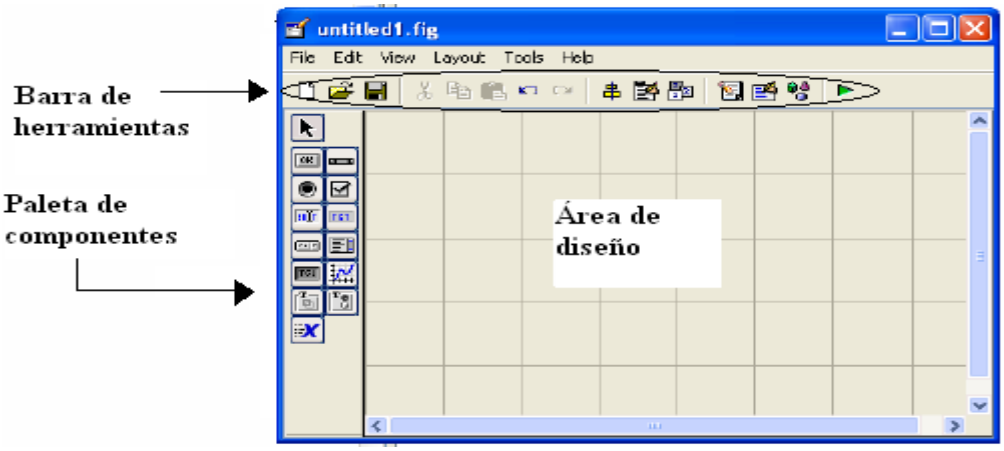


Figura 34*: Entorno de diseño de GUI*

- **Área de diseño.**

Se trata del espacio en el que se trabajará con los distintos componentes que se quieran incluir en el posterior archivo *.fig*. En él se colocarán los objetos, se alinearán o se determinará el tamaño de los mismos, así como el de la ventana en concreto.

Se podrá tener asimismo una primera impresión del resultado final del trabajo realizado.

- **Barra de herramientas**

Cuenta con las siguientes herramientas, identificadas por sus correspondientes iconos *(Figura 35):*

	Alinear objetos.
	Editor de menú.
	Editor de orden de etiqueta.
	Editor del M-file.
	Propiedades de objetos.
	Navegador de objetos.
	Grabar y ejecutar (ctrl. + T).

Figura 35: Elementos de la barra de herramientas y símbolos que los representan

- **Alinear objetos *(Align objects)***

Mediante esta herramienta se podrán alinear los distintos objetos situados en el área de diseño mediante distintos criterios, tales como la alineación horizontal, vertical... entre otras. Es tan sencillo como seleccionar los objetos que se quieren alinear y ejecutar la mencionada herramienta.

- **Editor de menú *(Menu editor)***

Se emplea para gestionar los menús que se quiere que aparezcan en la interfaz gráfica, pudiendo añadir, borrar, agrupar, renombrar y modificar cada uno de ellos.

- **Editor de orden de etiqueta *(Tab order editor)***

Ajusta la lengüeta y el orden de apilamiento de los componentes existentes en el diseño con el que se esté trabajando.

- **Editor del M-file *(M-file editor)***

Permite ejecutar secuencialmente un programa (en orden de líneas) que recoja múltiples instrucciones. Ideal para depurar programas y detectar fallos de programación de un modo más sencillo.

- **Propiedades de los objetos *(Property inspector)***

Mediante este icono se accede al *Property Inspector*, mediante el cual se pueden variar las propiedades de cada uno de los objetos, las cuales serán detalladas más adelante.

También se puede acceder al mismo pulsando con el botón derecho del ratón sobre el elemento y seleccionando la opción *Inspect Properties. (Figura 36)*

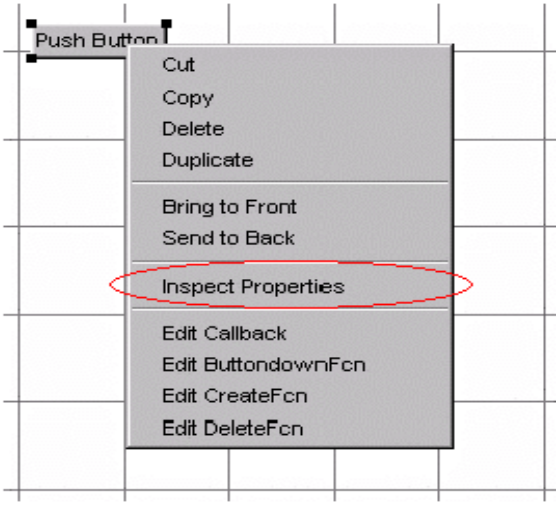


Figura 366: Acceso al Inspector de propiedades

- **Navegador de objetos *(Object browser)***

Ofrece un resumen de los objetos incluídos en la guide en forma de lista, de modo que se puedan localizar y gestionar de manera intuitiva y sencilla.

- **Grabar y ejecutar *(Run figure)***

Salva y ejecuta el GUI con el que se esté trabajando.

4.4.4.- Tratamiento y programación de ficheros *.m*

4.4.3.1.- Introducción

Una vez finalizado el diseño en cuanto a número de componentes y propiedades de los mismos que se quiere que aparezcan en la interfaz gráfica de usuario, al salvar los resultados se generará un archivo de texto automáticamente, con extensión *.m.*

El archivo *.m* que se crea tiene una estructura predeterminada. Consta de un encabezado y a continuación viene el código correspondiente a las siguientes subrutinas.

Esto se entenderá mejor poniendo un ejemplo, como puede ser una aplicación cuyo archivo *.fig* consista en 3 botones, un gráfico y un cuadro de edición.

Se generará un archivo *.m* con una estructura inicial como la siguiente:

```
function varargout = untitled1(varargin)
% UNTITLED1 Application M-file for untitled1.fig
%    FIG = UNTITLED1 launch untitled1 GUI.
%    UNTITLED1('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 20-Aug-2002 19:57:33

if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % Use system color scheme for figure:
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);

        guidata(fig, handles);

        if nargout > 0
                varargout{1} = fig;
        end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end

end
```

handles, Puntero a todas variables y elementos de la aplicación

guidata(), comando para guardar las variables de la aplicación

```
%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.
```

Subrutina Boton 1

```
% --------------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton1.
disp('pushbutton1 Callback not implemented yet.')
```

Subrutina Boton 2

```
% --------------------------------------------------------------------
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton2.
disp('pushbutton2 Callback not implemented yet.')
```

Subrutina Boton 3

```
% --------------------------------------------------------------------
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton3.
disp('pushbutton3 Callback not implemented yet.')
```

Subrutina Cuadro de Edición 1

```
% --------------------------------------------------------------------
function varargout = edit1_Callback(h, eventdata, handles, varargin)
```

```
% Stub for Callback of the uicontrol handles.edit1.
disp('edit1 Callback not implemented yet.')
```

Todo éste código ha sido generado automáticamente por MATLAB al salvar la guide diseñada tan sólo gráficamente por el momento.

Conviene detenerse en la siguiente instrucción, perteneciente al encabezamiento de la subrutina de, por ejemplo el botón 1

```
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
```

Por defecto, MATLAB otorga nombres a los elementos, en éste caso se trata de *pushbutton1*, correspondientes al valor de estilo del elemento en cuestión (*pushbutton*) con subíndice correspondiente al orden en el que se han ido agregando los elementos (el segundo botón está identificado como *pushbutton2* y así sucesivamente).

Es muy útil para la correcta programación y la minimización de confusiones que se edite éste valor, otorgándole un nombre identificativo de la función que vaya a cumplir el elemento en cuestión, como por ejemplo, *boton_ok*, ya que así será mucho más sencillo identificar a la subrutina correspondiente de un sólo vistazo, debido a la asociación inmediata que conlleva un nombre representativo.

Quizás en un principio se piense que se puede subsistir sin la edición de éstos valores, pero a medida que se incremente el número de elementos presentes en la interfaz gráfica, también lo hará la dificultad para la identificación de los mismos en el archivo con extensión *.m.*

4.4.3.2.- Manejo de datos entre los elementos de la aplicación y el archivo .m

Todos los valores de las propiedades de los elementos incluídos en la GUI (como por ejemplo el color, el valor, la posición, su string, etc.) y los valores de las variables transitorias del programa se guardan en una estructura de tipo MATLAB, capaz de almacenar matrices, strings, arreglos, vectores, etc., a los que se accederá mediante un único y mismo puntero para todos ellos [9].

El nombre del puntero se asigna en el encabezado del archivo *.m* , tomando por ejemplo el programa listado anteriormente el puntero se asigna en

```
handles = guihandles(fig);
```

Lo que implica que *handles*, es el puntero a los datos generado por MATLAB de la aplicación que se está diseñando.

Esta definición de puntero es salvada con la siguiente instrucción:

```
guidata(fig, handles);
```

Por lo que *guidata*, es entonces la función para salvar los datos de la aplicación en el puntero de datos *handles*.

Importante: *guidata* es la función que guarda las variables y propiedades de los elementos en la estructura de datos de la aplicación, lo que conlleva que, como regla general en cada subrutina se debe escribir en la última línea lo siguiente:

guidata(gcbo,handles);

Esto nos garantiza que cualquier cambio o asignación de propiedades o variables quede salvado.

Por ejemplo, si dentro de una subrutina una operación dio como resultado una variable *fi* para poder utilizarla desde el programa u otra subrutina debemos salvarla de la siguiente manera:

handles.fi=fi;
guidata(gcbo,handles);

La primera línea crea la variable *fi* a la estructura de datos de la aplicación apuntada por *handles* y la segunda graba el valor.

4.4.3.3.- Instrucciones get y set

- **Instrucción *get***

Se trata de una instrucción con la cual se podrán leer los valores introducidos por pantalla por el usuario, por ejemplo a través de un *editable text* y guardarlos en una variable para su posterior tratamiento [10].

Quizás sea preciso llevar a cabo alguna transformación de tipo de variable antes de memorizarlo para que los datos se puedan tratar correctamente a posteriori.

Para entenderlo mejor, se recurre al siguiente ejemplo:

muestras=str2double(get(handles.muestras,'String'));

Mediante *get* se lee el valor de la casilla en la que se escribe el número de muestras que desea el usuario; a continuación, mediante *str2double*, se transforman los datos de tipo string a tipo double y se memorizan en la variable muestras.

Si no se hubiera cambiado la identificación del elemento tipo *editable text*, habría que haber escrito:

muestras=str2double(get(handles.editabletext1,'String'));

Con lo que se habría complicado mucho la identificación del elemento de la interfaz gráfica de usuario.

- **Instrucción *set***

Es el comando mediante el cual se puede dar un valor determinado a un campo al desencadenarse un determinado evento, que puede ir desde la pulsación de un botón a el proceso correspondiente a la ejecución de una determinada subrutina [10].

También puede darse el caso expuesto en el caso de la instrucción get y necesitar cambiar el formato de los datos, por lo que se procederá de la misma forma que en dicha situación.

```
set(handles.frecuencia,'String',num2str(100));
```

En éste caso se transforman los datos en formato numérico a una cadena de caracteres, y se da dicho valor *(100)* al campo *frecuencia*.

4.4.3.4.- Leer y salvar ficheros

- **Leer un fichero**

A continuación se muestra el código preciso para cargar un fichero de datos (.dat) procedente de previas mediciones de señales, cualesquiera que sean, y representarlo en una gráfica incluida en la interfaz de usuario con la que se está trabajando [9].

```
[filename, pathname] = uigetfile( ...
   {'*.dat', 'All Points-Files (*.dat)'; ...
   '*.*','All Files (*.*)'}, ...
   'Load data');
% If "Cancel" is selected then return
if isequal([filename,pathname],[0,0])
   return
   % Otherwise construct the fullfilename and Check and load the file.
else
   File = fullfile(pathname,filename);
   datos=load('-ASCII',File);
   axes(handles.axes1);
   cla;
   plot(datos);
end
```

Notar que la etiqueta identificativa de la gráfica en la que se va a representar el fichero de datos *.dat* es el valor que MATLAB le otorga por defecto, es decir, *axes1*.

- **Salvar un fichero**

En esta ocasión se busca guardar una serie de datos obtenidos en tiempo real mediante la aplicación Btool en un fichero de datos en el que queden registrados para un posterior estudio con extensión *.dat* [9].

```
[filename, pathname] = uiputfile( ...
   {'*.dat', 'All Points-Files (*.dat)'}, ...
   'Save Starting point');

% If "Cancel" is selected then return.
if isequal([filename,pathname],[0,0])
   return
   % Otherwise construct the fullfilename and Check and load the file.
else
   File = fullfile(pathname,filename);
   save(File,'-ASCII','datos');
end
```

Posteriormente se podrá recuperar dicho fichero para representarlo en una gráfica mediante el procedimiento que se detalla en el apartado *Leer un fichero*.

4.4.3.5.- Abrir una nueva ventana

Esta opción es muy importante en MATLAB GUI, pues, por norma general, al llevar a cabo el desarrollo de una interfaz de usuario gráfica, se suele llamar a otras aplicaciones desde la ventana correspondiente a la aplicación principal [10].

Se consigue de éste modo ir subdividiendo el trabajo a realizar entre distintas aplicaciones dependientes de la principal, de modo que ésta llame a unas u otras en función de los requerimientos del usuario de la interfaz.

Es por ello por lo que es muy común que, desde el menu de acceso y habiendo seleccionado una determinada opción o pestaña del mismo, se precise de una instrucción que llame a la función correspondiente a la opción seleccionada.

Esto es debido a que, si se tratara de integrar todas las funcionalidades de una aplicación en tal sólo un documento *.m*, se elevaría la cantidad de errores en la programación y la posibilidad de detectarlos y solucionarlos exponencialmente.

Todo lo descrito anteriormente se lleva a cabo con tan sólo una línea de código, con lo que la sencillez de la programación de la opción de abrir una nueva ventana es determinante a la hora de considerarlo como una muy buena opción a la hora de la programación de interfaces gráficas con *MATLAB GUI*.

Se muestra tomando como ejemplo una llamada desde la ventana principal a un par de archivos, de extensiones *Procesado.m* y *Procesado.fig* respectivamente.

```
answerpc = Procesado();
```

Al pulsar el botón o seleccionar la opción que active este apartado del código del programa principal, se llamará a *Procesado*, abriéndose dicha aplicación en una nueva ventana.

4.4.4.- Ejemplo de programación de Guides

Para una mejor comprensión de los conceptos expuestos en el apartado 4.4.3. y dada la complejidad de asimilación que suponen, se adjunta a continuación un ejemplo simple con el que entender mejor todo lo que supone la programación de archivos *.m.*: El ejemplo de una sumadora.

Es más sencilla de entender la programación de archivos .fig, ya que el entorno de trabajo es mucho más intuitivo y amigable para el diseñador, por lo que se explicarán los elementos que es preciso incorporar en el área de trabajo y después se expresará como programar cada uno de los elementos precisos para que respondan acorde con las exigencias que el diseño requiere.

Se seguirán los pasos indicados en el *apartado 4.4.2* para llegar al entorno de trabajo, y una vez ahí, se colocarán los siguientes elementos *(Figura 37):*

- Dos componentes tipo *editable text* para introducir los sumandos
- Dos componentes tipo *static text* para representar el signo + y expresar el resultado de la operación.
- Un elemento del tipo *push button*, para llevar a cabo la operación

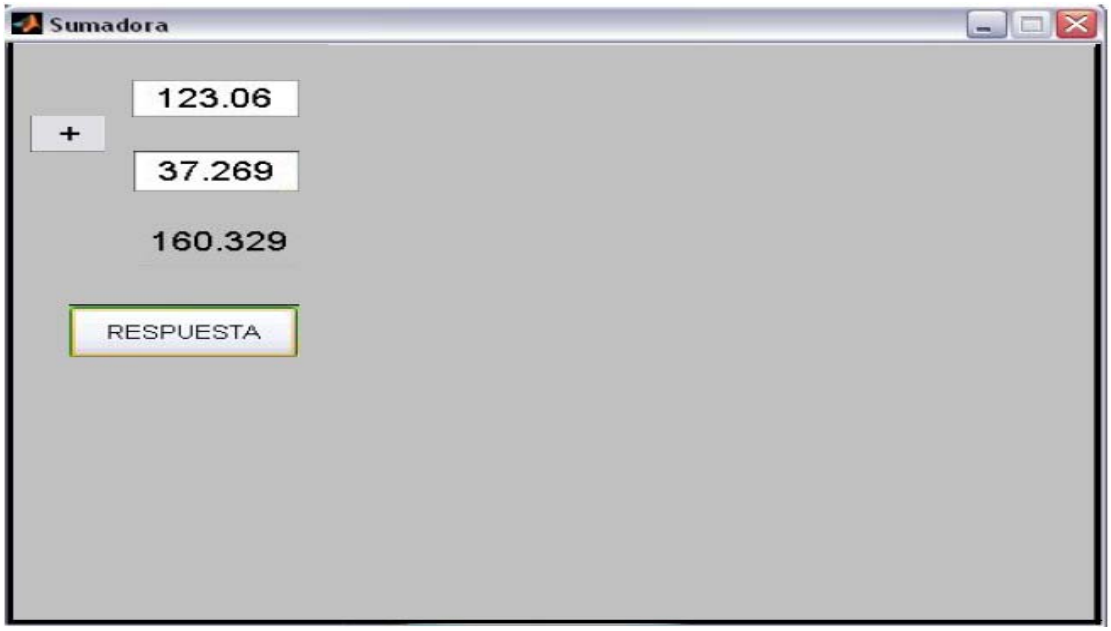


Figura 37: Aspecto de la sumadora

A continuación se adjunta el código completo correspondiente al desarrollo de la sumadora una vez finalizado, incluyendo tanto el código generado por MATLAB GUI como el que se ha de añadir para el correcto funcionamiento de la misma.

En él, se puede comprobar que cada elemento añadido tiene un apartado del código que MATLAB GUI reserva para poder programar las funcionalidades del mismo.

```
function varargout = Sumadora(varargin) %Inicialización de la aplicación por parte de
MATLAB
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @Sumadora_OpeningFcn, ...
                   'gui_OutputFcn', @Sumadora_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback', []);
if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
        gui_mainfcn(gui_State, varargin{:});
end
function Sumadora_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = Sumadora_OutputFcn(hObject, eventdata, handles)
        varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
        A=handles.edit1; %Se toma el valor introducido como primer sumando
        B=handles.edit2; %Se toma el valor introducido como segundo sumando
        ANSWER=A+B; %Se suman ambos valores
        set(handles.text2,'String',ANSWER); %Se expresa en la casilla de texto estático el
resultado

function edit1_Callback(hObject, eventdata, handles)
        Val=get(hObject,'String'); %Se almacena el valor ingresado
        NewVal = str2double(Val); %Se transforma a formato double
        handles.edit1=NewVal; %Se almacena en el identificador
        guidata(hObject,handles); %Se salvan los datos de la aplicación

function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), %Propiedades del elemento
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
        Val=get(hObject,'String'); %Almacenar valor ingresado
        NewVal = str2double(Val); %Transformar a formato double
        handles.edit2=NewVal; %Almacenar en identificador
        guidata(hObject,handles); %Salvar datos de la aplicación

function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), %Propiedades del elemento
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end
```

CAPÍTULO 5: DESCRIPCIÓN DE LA APLICACIÓN BTOOL

5.- DESCRIPCIÓN DE LA APLICACIÓN BTOOL

5.1.- Introducción: proceso de diseño de una GUI

Antes de empezar a programar es imprescindible hablar con el usuario final de la GUI, debido a que es básico comprender cuáles son las necesidades exactas que tiene que cubrir la aplicación a diseñar.

Para conseguir este fin es necesario entender el tipo de datos y variables que son introducidas por el usuario, así como las posibles excepciones que puedan tener lugar en determinadas situaciones.

Es preciso también conocer el modo en el que el cliente quiere que se presenten los datos resultantes del funcionamiento de la aplicación a diseñar.

La parte de diseño es, con mucha diferencia, la más importante desde el punto de vista del usuario y por tanto también lo será desde el punto de vista empresarial, ya que éste factor es el que moviliza toda la actividad de implementación y mejora de las aplicaciones mediante una interfaz gráfica de usuario.

Para diseñar correctamente una GUI se suele proceder mediante un primer borrador, por lo general realizado a mano alzada sobre un papel, en el que representar un boceto con el cual explicar al cliente las ideas que se tienen para llevar a cabo en el proyecto, escuchando las suyas propias, dando lugar a una realimentación de conceptos y puntos de vista que puede llevar al cambio de las ideas preconcebidas en un principio por el diseñador.

De esta forma se consigue la minimización de los imprevistos en el desarrollo de la herramienta de todo tipo, ya sea por imposibilidades derivadas de objetivos inalcanzables o encarecimiento económico o temporal del proyecto.

Se consigue además que el cliente se sienta parte activa del proyecto y lo sienta como propio, ya que en él estarán depositada parte de sus ideas y preferencias, aumentando de este modo la satisfacción final con el resultado obtenido.

Una vez que se tienen claras las especificaciones y objetivos perseguidos a satisfacer por la interfaz de usuario a desarrollar, es necesario hacer un programa tipo "script" con igual funcionalidad que la GUI que se quiere diseñar.

Antes de incorporar el programa a la aplicación es necesario hacer todo tipo de pruebas con él, planteando casos fuera de lo usual y poniendo a prueba al programa buscando el máximo de estabilidad posible hasta que se esté completamente seguro de que el programa que se va a incorporar a la GUI es el adecuado.

Una vez que se tenga el "script" guardado se podrá incorporar a la herramienta, de modo que al hacer pruebas con la misma, se pueda comprobar que los resultados obtenidos con el mismo se corresponden con los que ofrece la aplicación.

5.2.- Descripción de las ventanas que constituyen la aplicación

5.2.1.- Ventana principal

Se trata de la ventana que constituye el punto de partida tras ejecutar la aplicación desde la que se podrá acceder a todas las funcionalidades que ofrece la herramienta (*Figura 38*), subdivididas en cuatro pestañas que comprenden cuatro bloques temáticos en cuanto a tratamiento de datos y adquisición de los mismos se refiere, que son *Cargar*, *Adquisición de datos, Transformadas y Defectos.*

Alguna de ellas da acceso a una ventana directamente desde la cual llevar a cabo el cometido deseado y otras, sin embargo, se despliegan dando lugar a varias opciones.

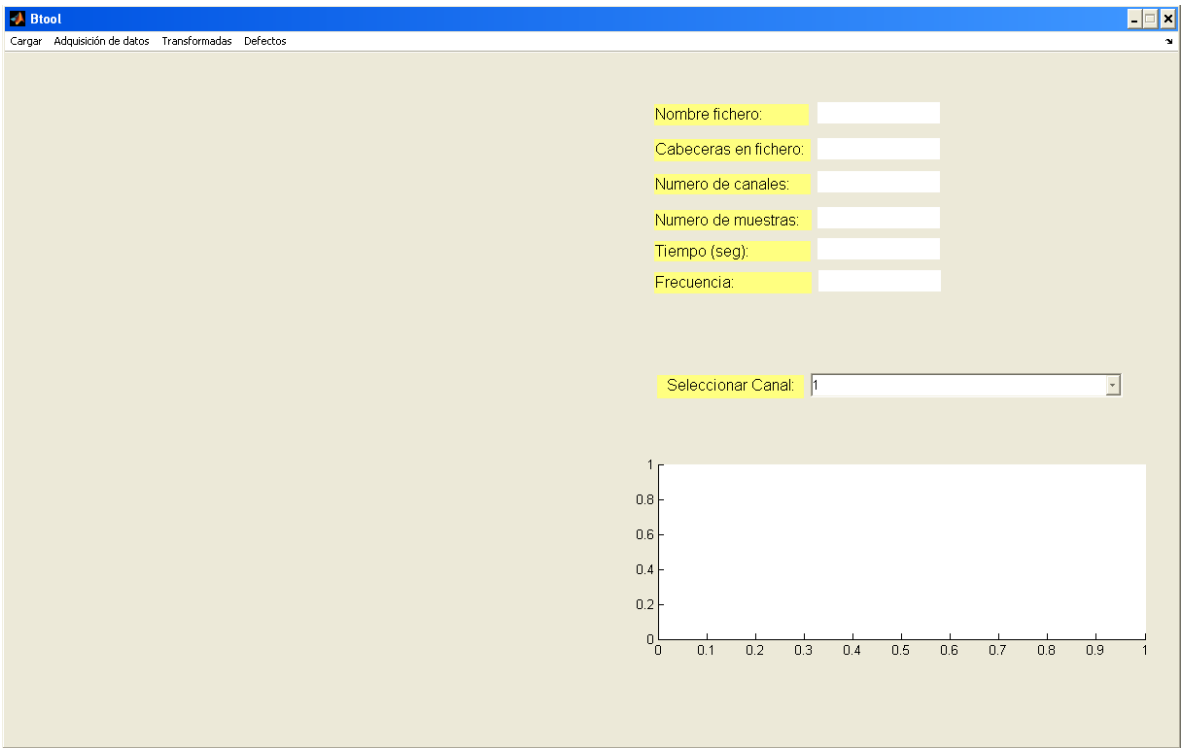


Figura 38: Ventana principal de la aplicación Btool

5.2.2.- Cargar

Es una ventana a la que se accede directamente mediante la pestaña *Cargar*, presente en la ventana principal que se acaba de describir. Permite tomar datos previamente medidos y guardados en el disco duro con el fin de representarlos y tratarlos.

Al hacer *click* en dicha pestaña se abrirá un cuadro de diálogo *(Figura 39)* mediante el cual se podrá seleccionar el dato que interese para representarlo en la gráfica desde la que se visualiza la señal original.

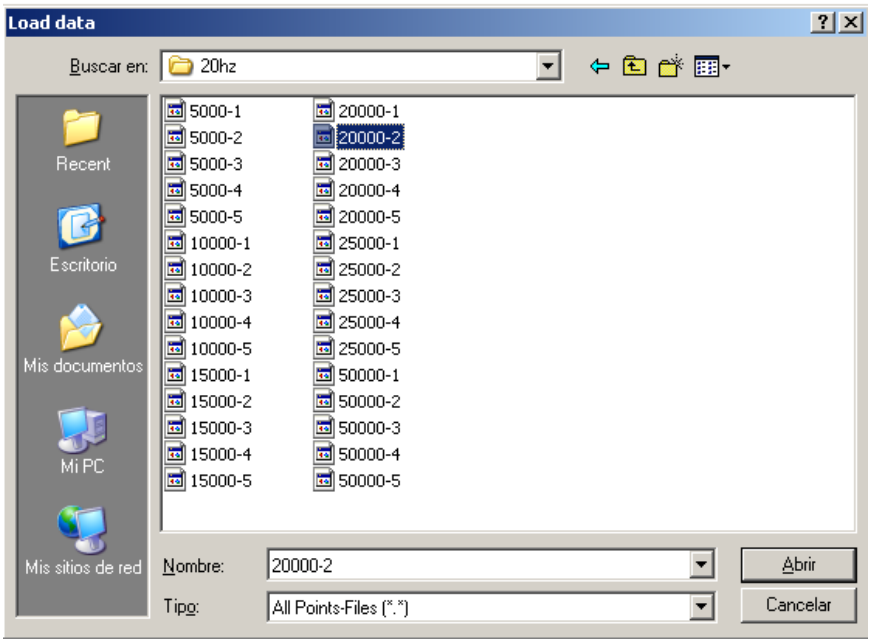


Figura 39: Cuadro de diálogo perteneciente al menú Cargar

Una vez seleccionada la señal con la cual se quiere trabajar, se pulsará el botón *Abrir* en la ventana correspondiente al menú *Cargar* (*Figura 39*) viéndose con ello representada dicha señal (*Figura 40*).

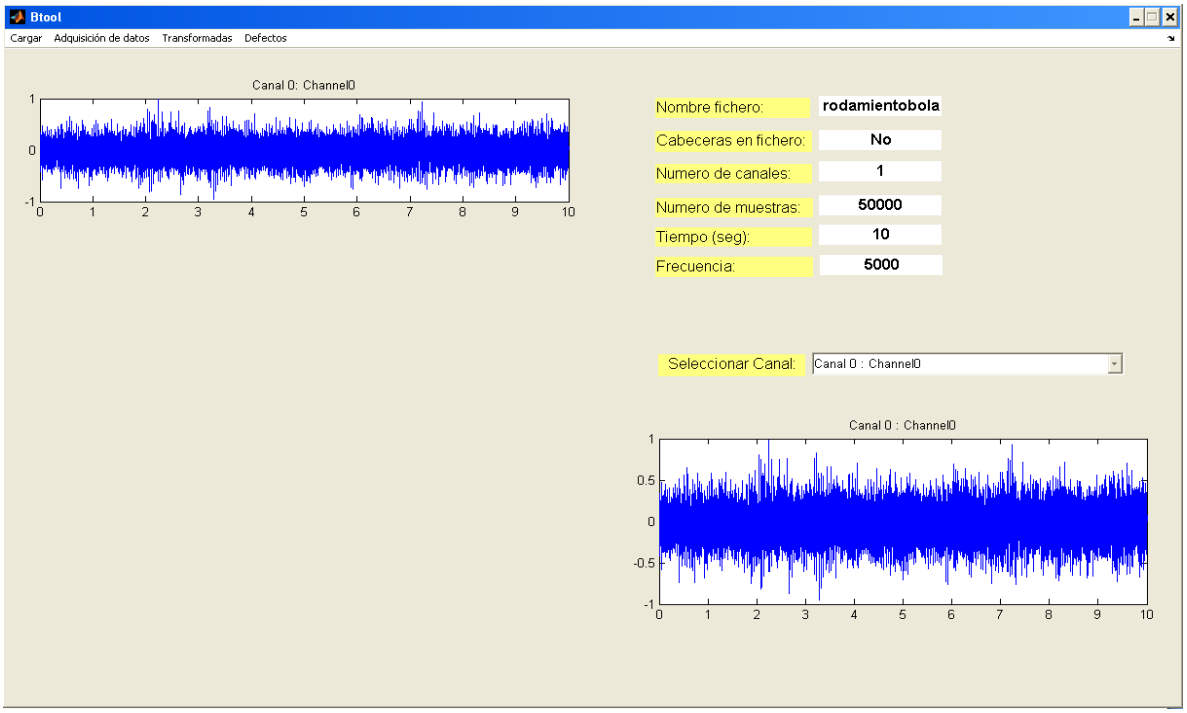


Figura 40: Señal previamente guardada cargada en la ventana Btool

5.2.3.- Select Channel

A esta ventana se accede a través de la pestaña *Adquisición de datos* situada en la ventana principal de la aplicación *Btool (Figura 38).*

Permite tomar datos seleccionando como fuente de los mismos un determinado canal a elegir entre los ocho que permite conectar la aplicación desde un panel desplegable. Tiene el aspecto que se muestra en la *Figura 41*.

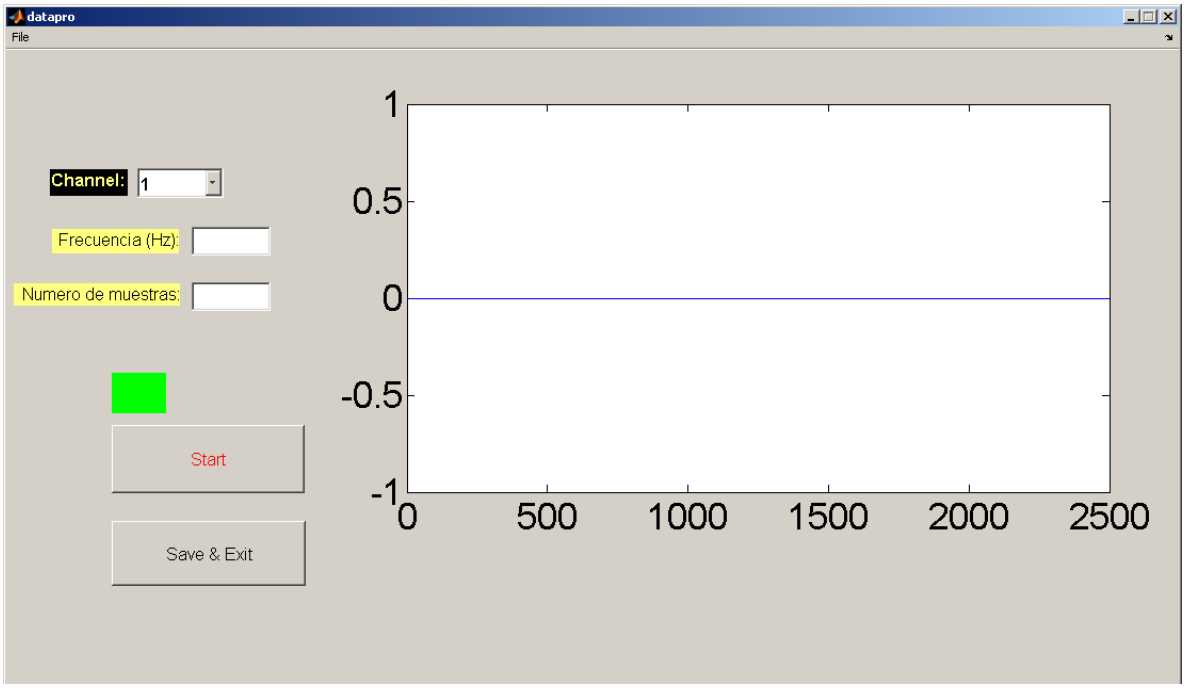


Figura 41: Ventana perteneciente a la aplicación Select Channel

Conviene hacer notar que, al ejecutar la ventana *Select Channel* (al igual que en el caso de ejecutar la ventana *All Channels*), el sistema detectará y comprobará si la tarjeta de adquisición de datos está conectada.

En caso de estar conectada, el sistema se preparará para la adquisición de datos en tiempo real, de la que se hablará más adelante; mientras que de no ser así aparecerá mensaje que se representa en la *Figura 42*, accediendo posteriormente al modo simulación de dicha ventana.

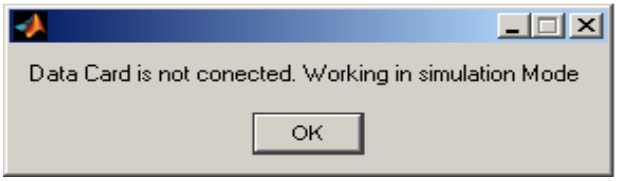


Figura 42: Tarjeta de adquisición de datos no conectada

En el modo de simulación se dibujará una señal sinusoidal por defecto al pulsar el botón *Start*. La frecuencia de dicha señal sinusoidal variará en función del canal que se represente como fuente de datos, para poder así representar distintas señales de gran utilidad, debido a su gran aplicación de manera inmediata para poder tratarlas en la aplicación *Btool*.

En la *Figura 43* se muestra la ventana *Select Channel* trabajando en modo simulación cuando se ha seleccionado el *canal 1* como proveedor de la información, si se seleccionara el *canal 2*, la frecuencia sería el doble de la del *canal 1*, y esa misma proporción también se dará para el resto de los canales, por lo que la frecuencia de la señal sinusoidal vendrá determinada por el número de canal que se seleccione en el modo simulación.

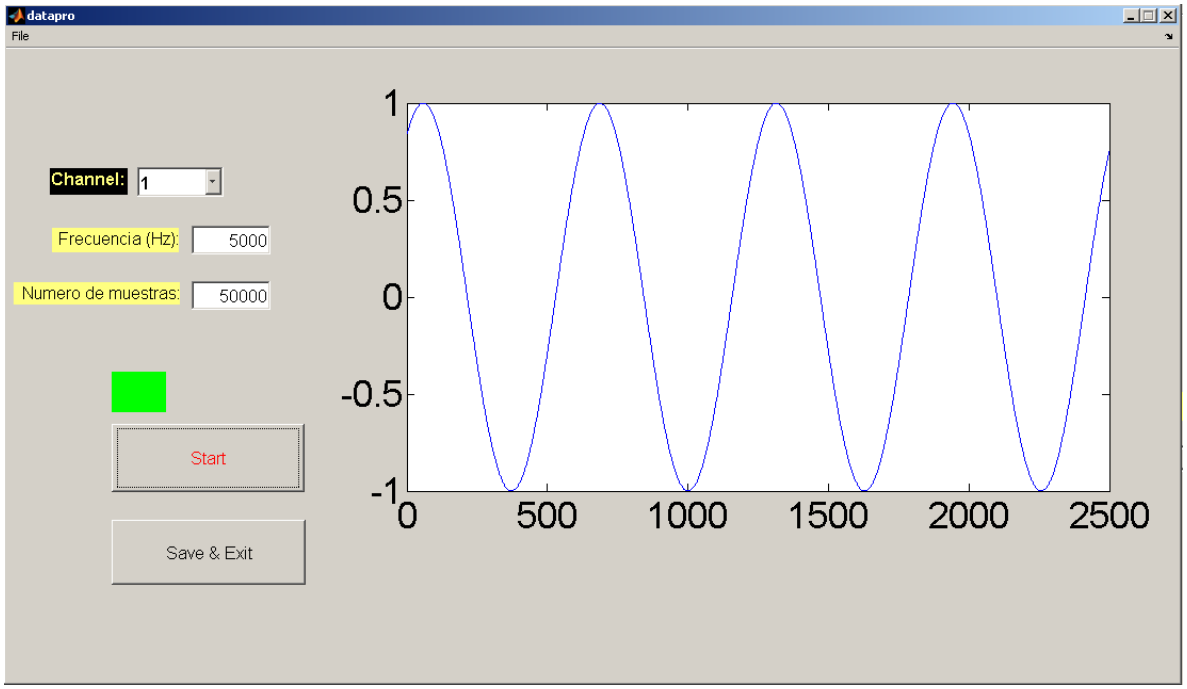


Figura 43: Aplicación Select Channel trabajando en modo simulación

Una vez finalizada la toma de datos, y con el objeto de transmitir los resultados a la ventana principal para su posterior tratamiento, se ha de presionar el botón *Save & Exit*, tras lo cual aparecerá un cuadro de diálogo pidiendo confirmación *(Figura 44):*

Figura 44: Petición de confirmación de cierre de la aplicación Select Channel

Tras confirmar, los datos tomados mediante la aplicación *Select Channel* pasarán a la ventana principal *(Figura 44).*

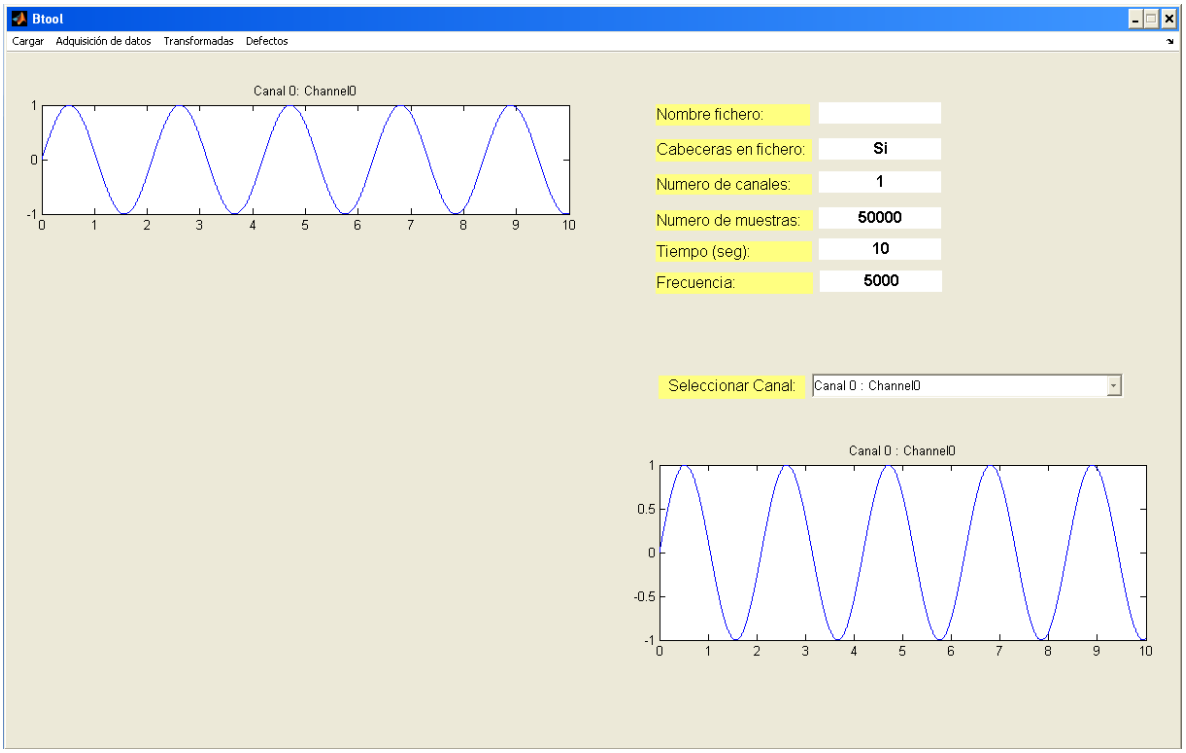


Figura 44: Datos tomados de Select Channel representados en la ventana principal.

5.2.4.- All Channels

La ejecución de esta aplicación se lleva a cabo a través de la ruta de acceso correspondiente a *Adquisición de datos/All channels.*

Se trata de una ventana que permite la toma de datos en tiempo real de hasta ocho canales simultáneamente, que se podrán identificar mediante el nombre de la fuente de datos de cada uno de ellos en el menú configuración (que se presentará más adelante) para una identificación más sencilla, pudiendo posteriormente representar aquellos que se consideren oportuno para el estudio a llevar a cabo en un igual número de gráficas, que se ordenarán y se repartirán el espacio disponible para la visualización de los datos en función del número de canales que se quieran representar.

A continuación se visualiza el aspecto inicial al ejecutar dicha ventana *(Figura 45)*, y posteriormente se irá desarrollando el modo de utilizar la herramienta *All channels* en todas sus extensas facetas.

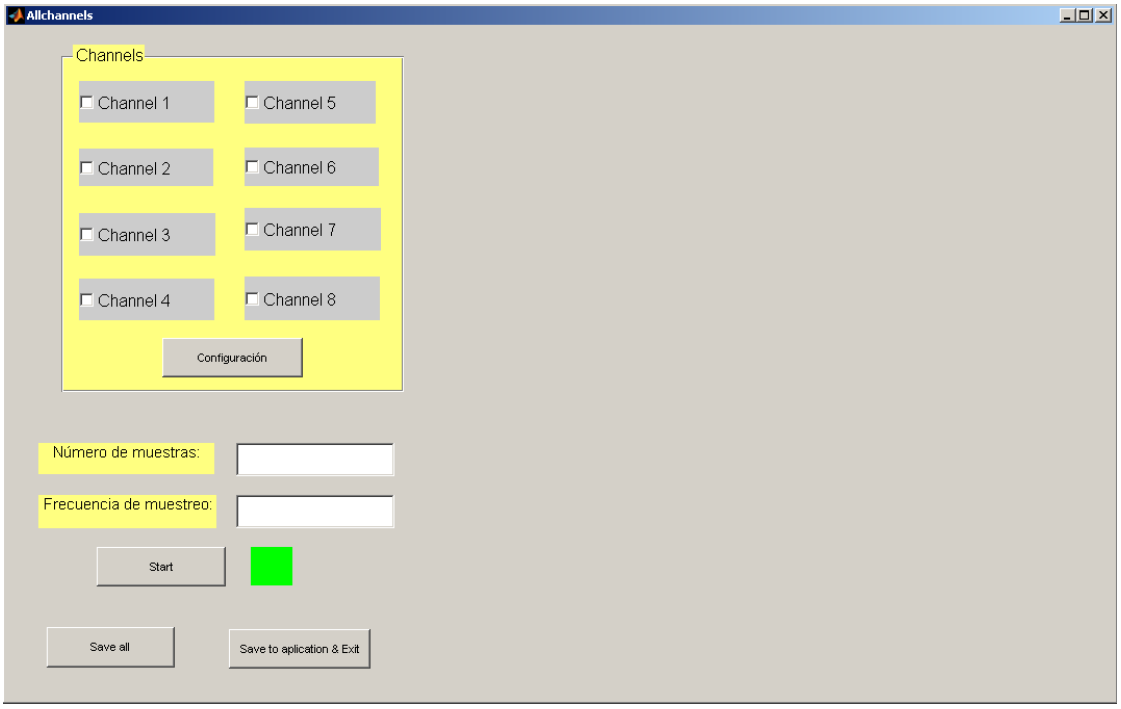


Figura 45: Ventana perteneciente a la aplicación All Channels

Mediante la ventana *All channels* se puede acceder a la edición de los nombres que se le otorga a cada uno de los canales, muy útil para una identificación rápida e intuitiva de las fuentes de información que se estén estudiando.

Para ello se ha de presionar el botón *Configuración*, situado en el panel en el cual se presentan los ocho canales que se pueden utilizar como fuente de datos para la aplicación.

Conviene hacer notar que, por defecto, las etiquetas que recibirán los mismos serán Channel 1, ... , Channel 8. Para editarlas, se accede al siguiente menú *(Figura 46)* del modo indicado anteriormente.

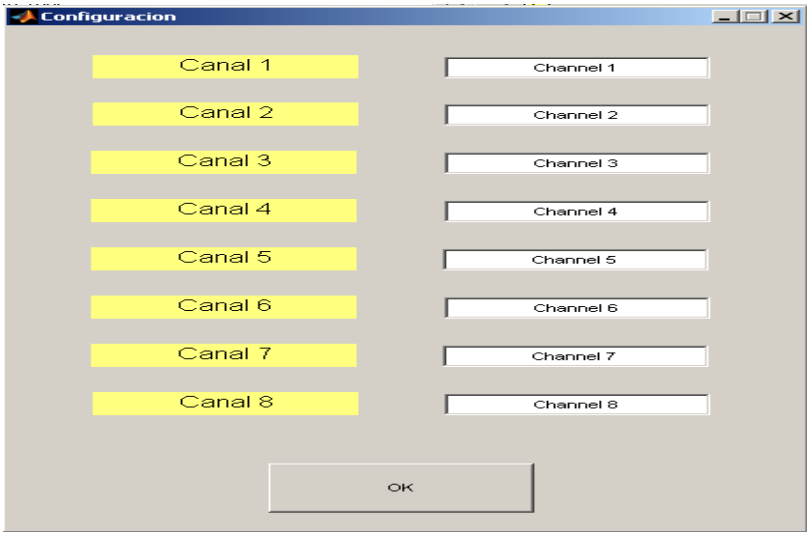


Figura 46: Cuadro de diálogo para la edición de las etiquetas de los canales

Para proceder a editar las etiquetas se actuará de un modo sencillo; basta con introducir mediante el teclado los nuevos nombres que se le quiera otorgar a cada uno de los canales, tal y como se procede en la *Figura 47*.

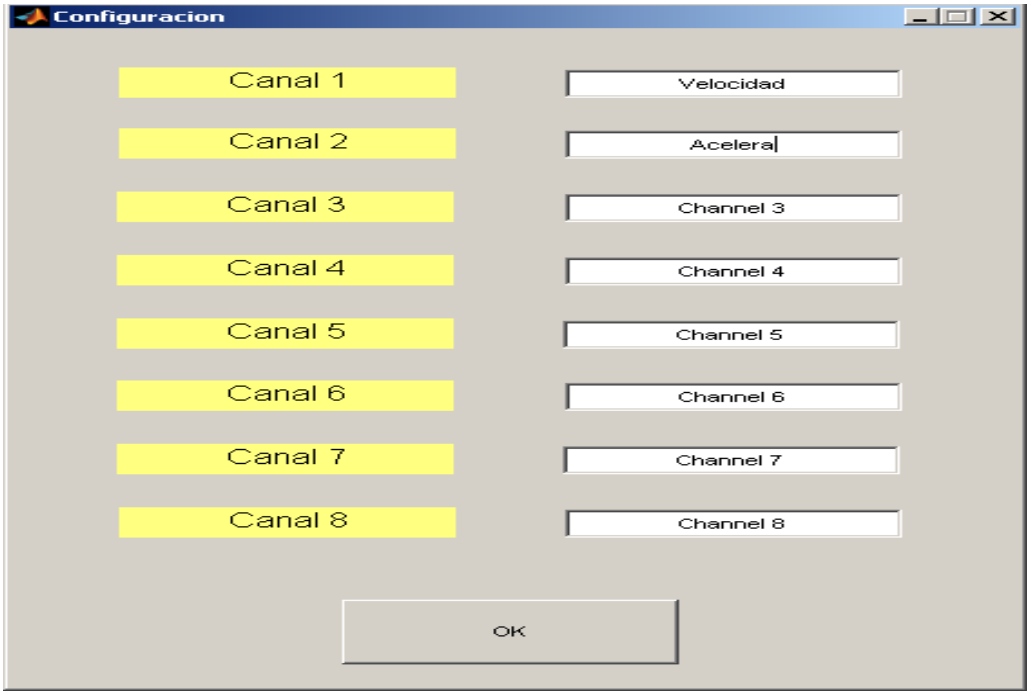


Figura 47: Escritura de nuevos valores de etiquetas de los canales

Una vez editadas las identificaciones de los canales y presionando el botón *OK* se pasará la información a la ventana *All Channels*, quedando del modo que se indica en la *Figura 48* la situación en la misma.

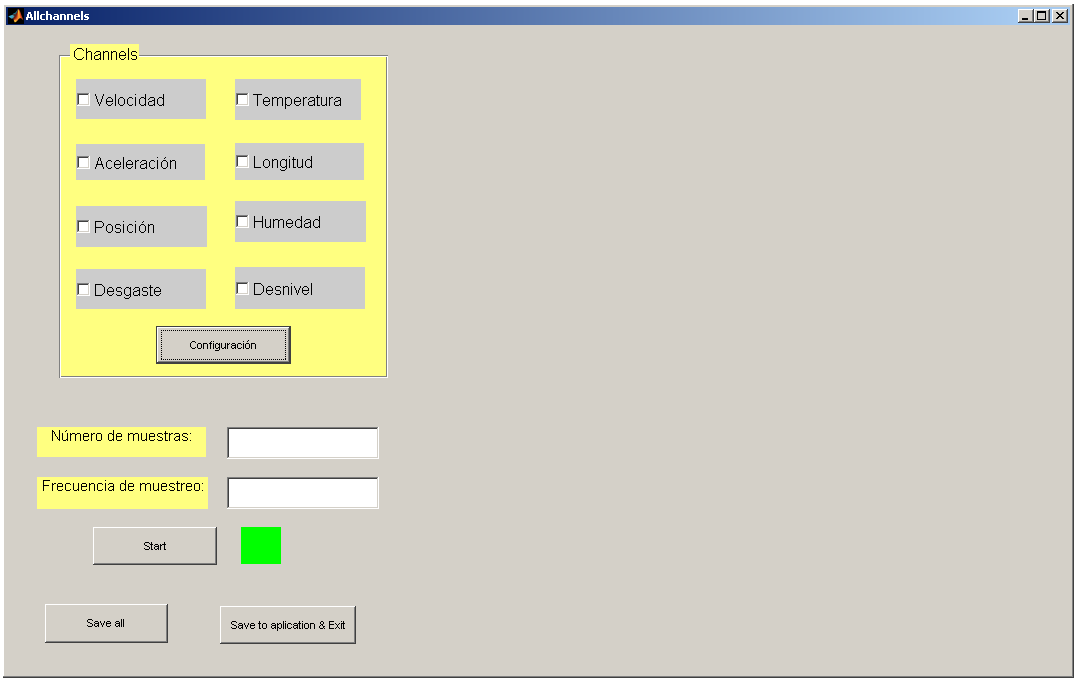


Figura 48: Estado de la aplicación All Channels tras la edición de sus etiquetas

Obsérvese la utilidad en cuanto a lo intuitivo de la posibilidad que ofrece la aplicación de poder identificar de forma sencilla a cada uno de los canales que entran en juego en la toma de datos que tendrá lugar posteriormente sin el más mínimo atisbo de error por confusión.

Se acompañará asimismo del nombre que se le ha otorgado a cada canal mediante una pequeña reseña a cada una de las gráficas que se quieran representar.

Una vez identificado cada canal, se pasará a la toma de datos propiamente dicha. Una vez más, la sencillez a la hora de interactuar con la aplicación se ha incrementado al máximo, pues para ello basta con seguir y llevar a cabo cuatro pasos elementales, que se pasan a detallar a continuación:

1.- Se seleccionan los canales que se quieran medir en el listado de los mismos.

2.- Se expresa el número de muestras que se quieran obtener para el estudio del comportamiento de dicho canal en el cuadro de diálogo que viene acompañado por la reseña *Número de muestras*.

3.- Se introduce por teclado la frecuencia de muestreo con la que se quiere llevar a cabo la medida de la fuente del canal a representar para el estudio del comportamiento del mismo en el cuadro de diálogo que viene acompañado por la etiqueta *Frecuencia de muestreo*.

4.- Se presiona el botón *Start*, dando así comienzo a la toma de datos y posterior representación de los canales que hayan sido previamente seleccionados de forma totalmente autónoma y automática.

En el caso que se presenta a continuación (*Figura 62*) se ha optado por medir y representar las variables *Velocidad*, *Temperatura* y *Humedad*, tomando quinientas muestras a una frecuencia de 5kHz (los nombres que se les ha otorgado a los canales para éste ejemplo son sólo orientativos, pudiendo tomar el nombre de cualquier otra variable no presente en el citado ejemplo).

Como se está trabajando sin la tarjeta de adquisición de datos ni el equipo conectados al PC, el programa actuará en modo simulación, ignorando los valores de *frecuencia* y *número de muestras* introducidos, como previamente se indicó, para representar las señales sinusoidales que el programa dibuja por defecto.

En tal caso, los canales seleccionados cambiarán su nomenclatura automáticamente para pasar a tomar los valores de las funciones sinusoidales que representan, acompañadas de la reseña que indica la magnitud de la frecuencia asociada a cada canal *(Figura 49).*

Si se quiere comprobar el comportamiento en sus aplicaciones prácticas, se invita a la consulta del *Apartado 6*, correspondiente a los resultados experimentales.

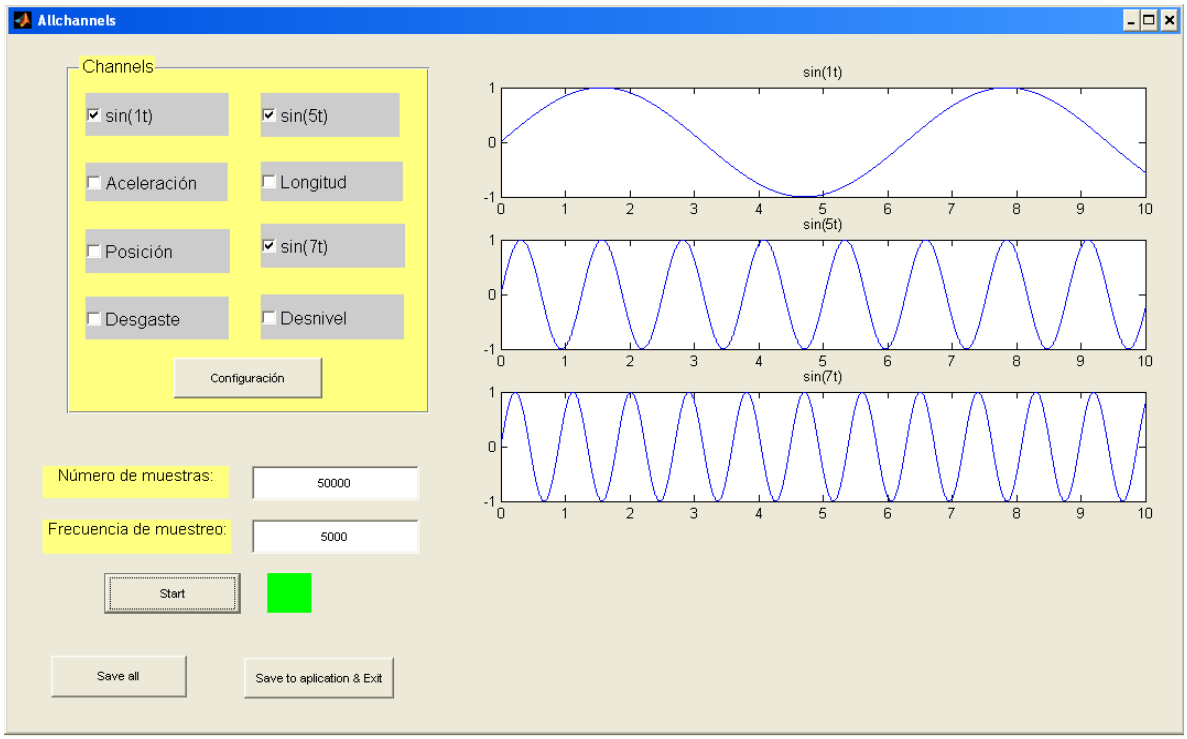


Figura 49: Representación de tres canales en All Channels en modo simulación

Se pasa a estudiar a continuación el caso en el que se quiere llevar a cabo una medida sometiendo al sistema a su máximo rendimiento, es decir, tomando datos de ocho canales simultáneamente.

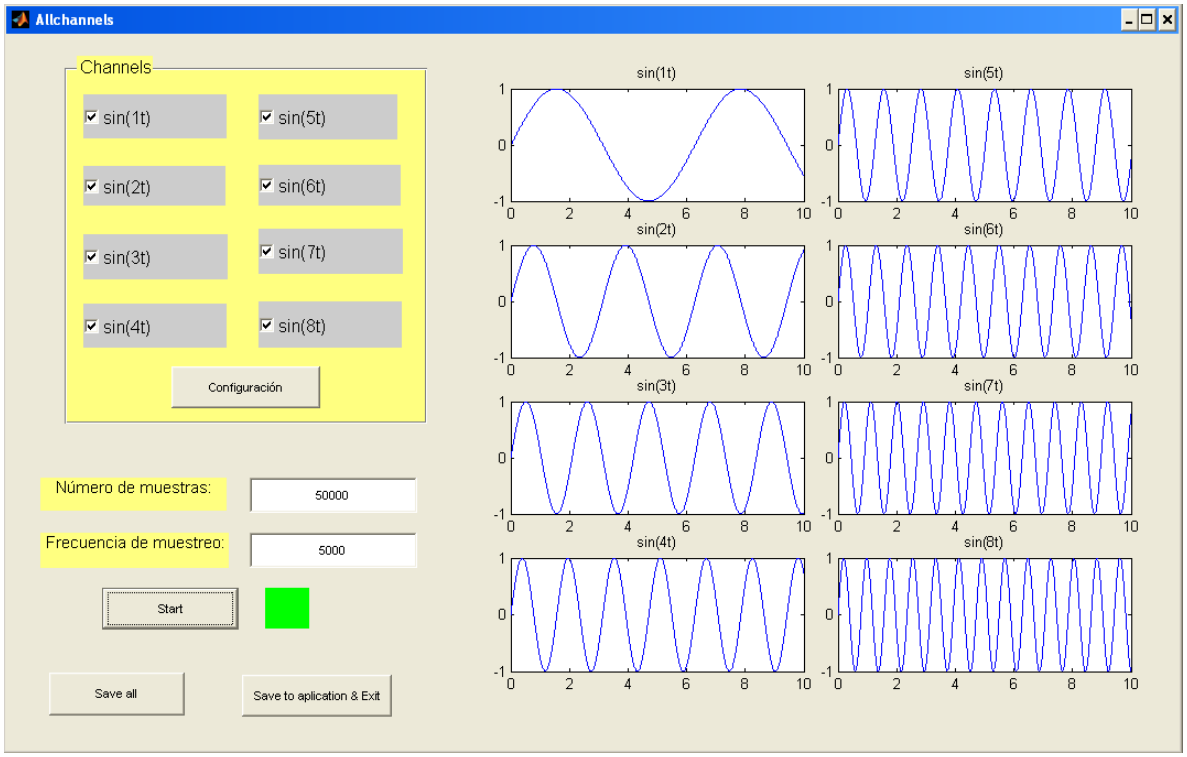


Figura 50: Representación de ocho canales en All Channels en modo simulación

Conviene hacer notar que en este caso, el sistema ejecutará un algoritmo interno que permitirá que se subdivida la zona de presentación de datos en gráficas de un tamaño más reducido que en el caso del ejemplo anterior *(Figura 50)*, con objeto de así poder representar la totalidad de los canales en el mismo espacio en el que en el citado ejemplo tan sólo se representaban tres.

Esto será así en todos los casos, por lo que el tamaño de las gráficas dependerá del número de canales que se quieran estudiar a cada momento, lo cual resulta básico a la hora de una correcta visualización de las medidas tomadas.

También conviene reseñar que las gráficas siguen el mismo orden que en el listado de canales y que, en caso de ausencia de alguna de ellas, dicho orden no se verá alterado, representando la gráfica correspondiente al siguiente canal seleccionado en orden ascendente.

Dentro del programa *All Channels* se presentan dos opciones para almacenar los datos tomados en tiempo real, que se detallan a continuación:

- *Save All:* Con esta opción se ordenará al programa que salve todos los datos pertenecientes a todos los canales seleccionados en un fichero de extensión *.dat* para un posterior estudio y recuperación de los mismos.

- *Save to aplication & Exit:* Pulsando el botón correpondiente a esta opción se procederá a la comunicación entre la ventana principal y la ventana perteneciente a la aplicación All Channels para transferir la información obtenida de una a otra, tal y como se observa en la *Figura 51*.

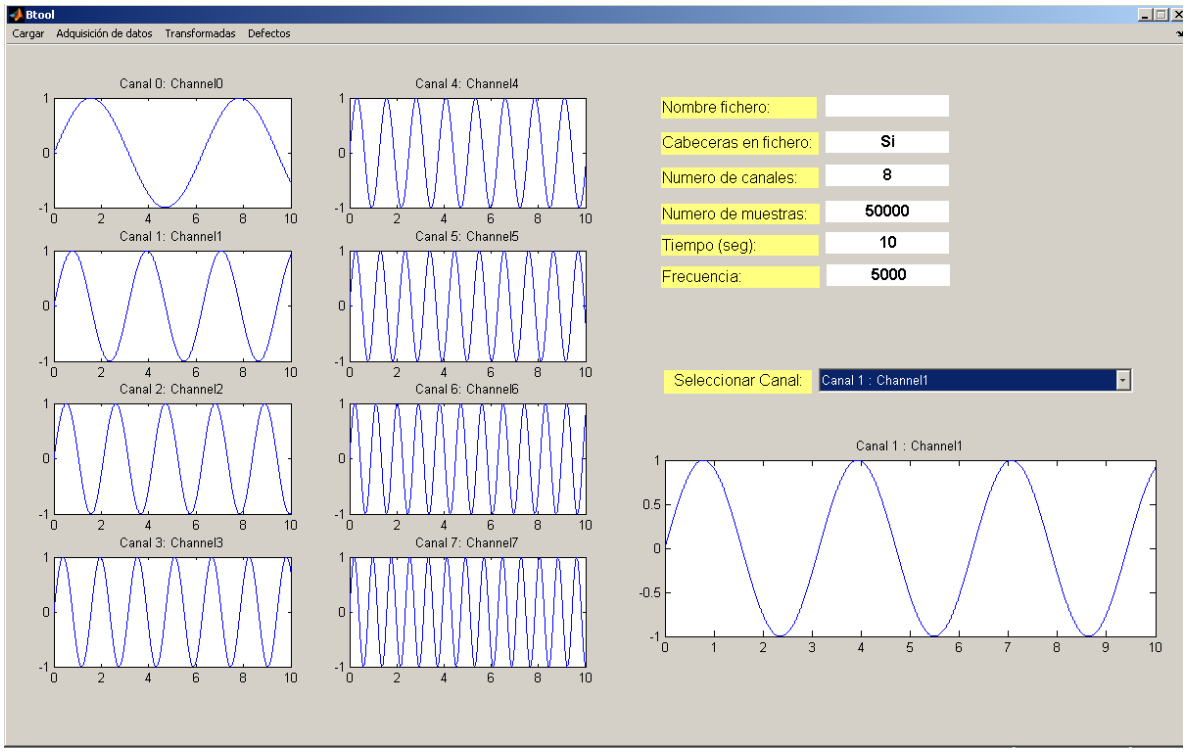


Figura 51: Datos adquiridos mediante All Channels representados en la ventana principal

5.2.5.- Espectro de frecuencias

Ésta opción ofrece la posibilidad de calcular el espectro de frecuencias de la señal que se está tratando en la ventana principal de la aplicación *Btool*.

No obstante, se puede obtener el espectro de cualquier otra señal que se esté tratando, lo que ofrece un abanico de posibilidades infinito a la hora de llevar a cabo estudio de señales.

Se accede a dicha ventana a través de la ruta *Transformadas/Espectro de frecuencias*.

El resultado de la operación aparecerá en una nueva ventana identificada con la etiqueta *Espectro de frecuencias*, en la que aparecerá el mismo en la gráfica identificada como Transformada; pero además se representará en otra gráfica, con etiqueta Señal el aspecto de la señal a la que corresponden el mismo y que está siendo objeto de estudio *(Figura 52)*.

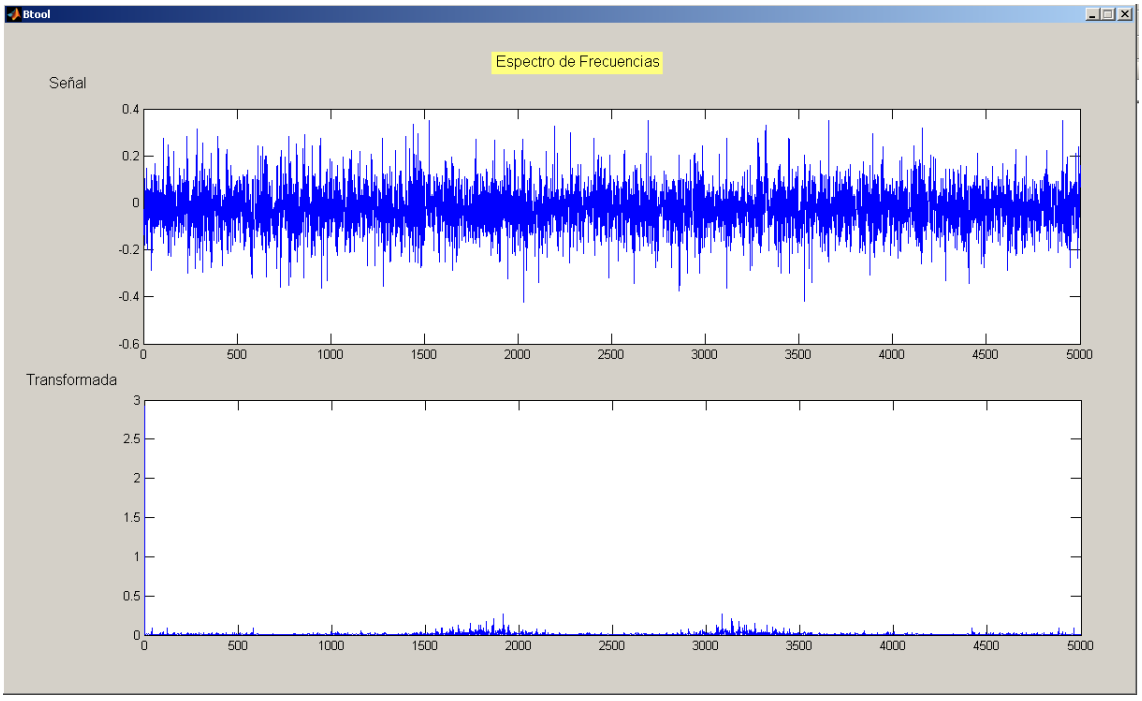


Figura 52: Espectro de frecuencias de una señal determinada

5.2.6.- Transformada de Hilbert

Ésta opción permite calcular la *transformada de Hilbert*, opción muy útil para el objetivo para el que está encaminado el presente proyecto, ya que es una herramienta que se puede aplicar como tratamiento previo a la posterior representación del espectro de frecuencias de la señal a tratar, ya que elimina componentes de frecuencia que

"ensucian" el estudio de las vibraciones mecánicas que podrían repercutir en la calidad de los resultados obtenidos.

Se ha de aplicar sobre una señal previamente cargada en la ventana principal, pudiendo proceder de la toma de datos en tiempo real, o bien de la recuperación de medidas correspondiente a estudios que se hayan realizado con anterioridad.

Se accede a la aplicación siguiendo la ruta de acceso desde la ventana principal correspondiente a *Transformadas/Hilbert.*

5.2.7.- Descomposición en Wavelets

En este apartado se va a tratar la obtención de los niveles de energía de la señal procesada y registrada mediante las *transformadas Wavelet*, aplicación diseñada y orientada expresamente para el estudio del mantenimiento predictivo mediante el análisis de señales vibratorias.

Se accede a la aplicación siguiendo la ruta desde la ventana principal marcada por *Transformadas/Descomposición de Wavelets.*

Según se accede a la misma el aspecto que presenta es el que se indica en la *Figura 53:*

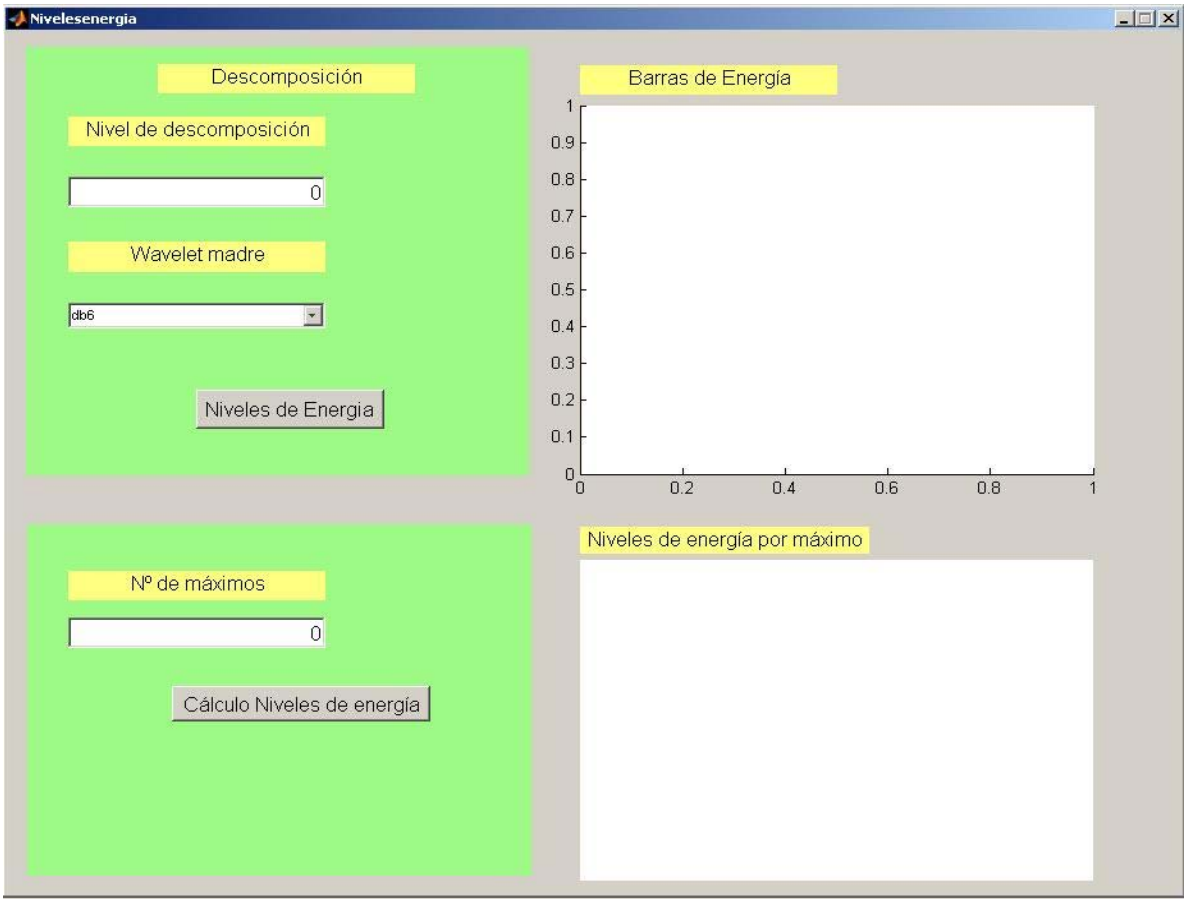


Figura 53: Aspecto de la aplicación Descomposición de Wavelets

Es una ventana subdividida en módulos con distintas funcionalidades, en los que el usuario podrá seleccionar y modificar los parámetros que a continuación se indican y detallan:

5.2.7.1.- Nivel de descomposición y transformada madre wavelet

Son dos parámetros que darán lugar al primer estudio de los dos que permite la aplicación que ocupa este apartado:

- El *nivel de descomposición* indica el número de veces que el sistema va a subdividir el espectro de la señal en partes iguales, representando posteriormente el nivel de energía que la misma posee en cada tramo en la gráfica correspondiente.

- La *transformada de wavelets madre* se podrá seleccionar en el desplegable que se muestra dentro de la Figura 53, y en él están implementadas todas las transformadas de wavelets madre que ofrece MATLAB.

En la *Figura 54* se muestra la descomposición del *Canal 1* de la *Figura 52* en dos niveles de energía, tomando como transformada madre una *Daubechies 6*.

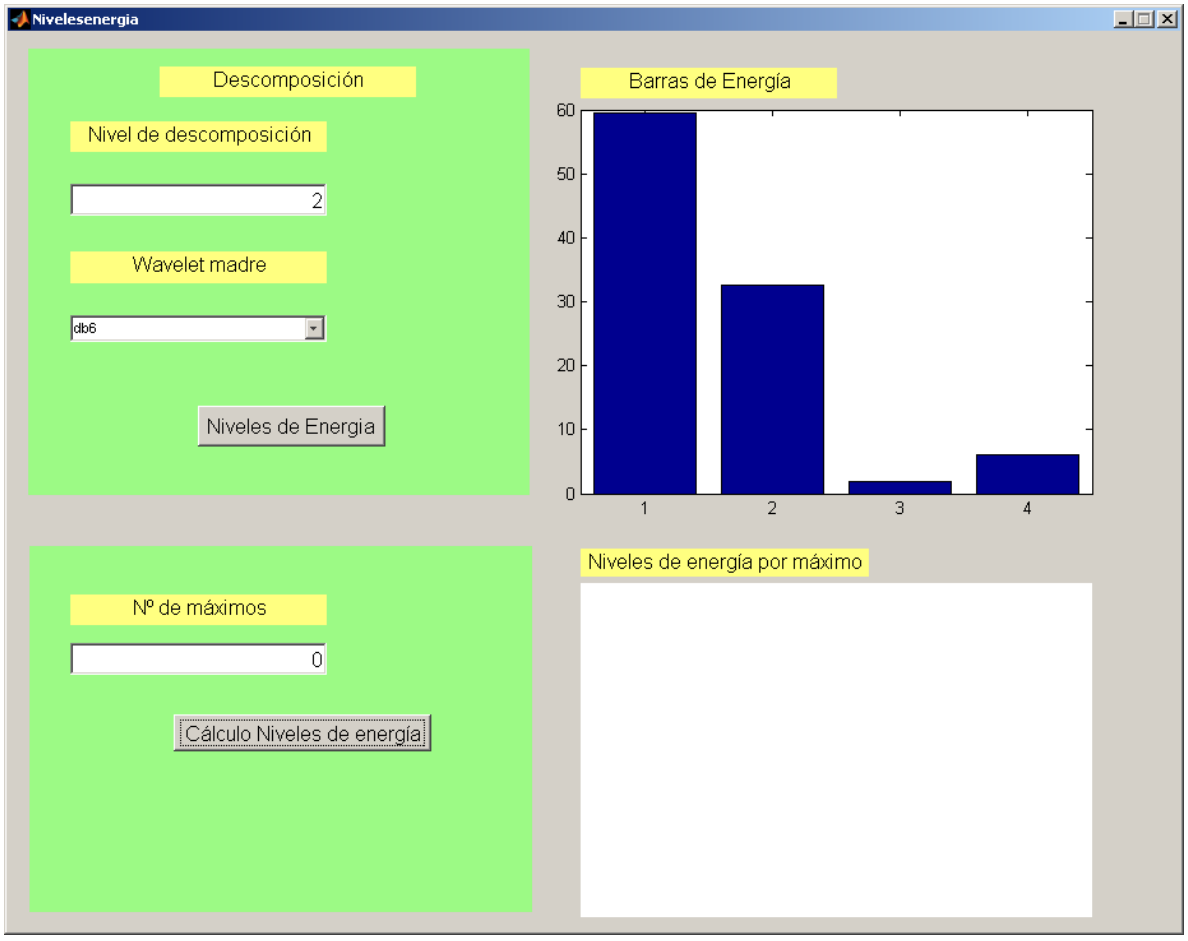


Figura 54: Bandas de energía de una función senoidal

Como complemento al estudio llevado a cabo, aparecerá una ventana nueva en la que se representará la señal que se esté tratando (en este caso una función senoidal) en los rangos de frecuencia en los que ha sido subdividida con el objeto de llevar a cabo un estudio más pormenorizado, tal y como se muestra en la *Figura 55.*

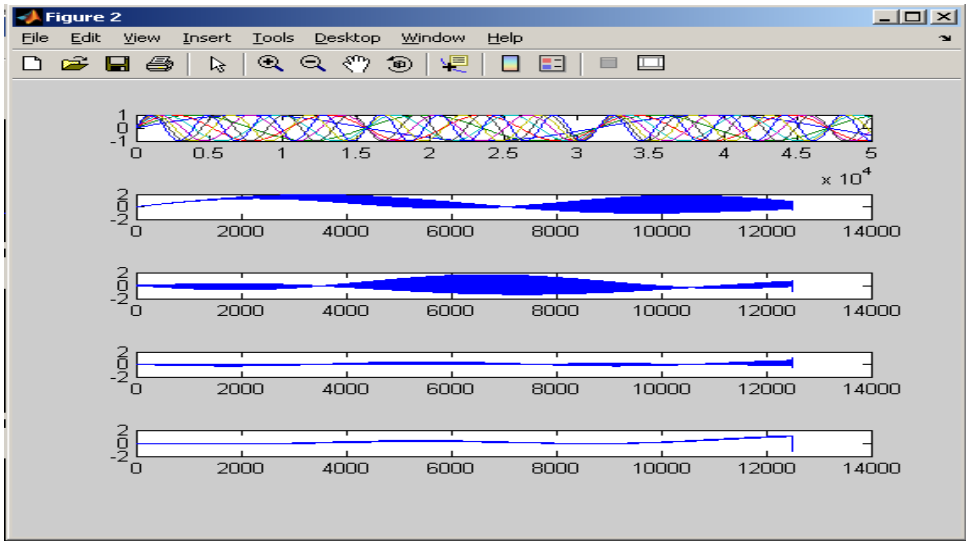


Figura 55: Rango frecuencial de una senoide subdividida en cuatro partes iguales

Se puede observar que la *frecuencia de muestreo* (5kHz) ha sido subdividida en cuatro rangos de 1250 Hz cada uno.

5.2.7.2.- Número de máximos de energía:

Este apartado es un complemento realmente interesante al cálculo de los niveles de energía. El usuario podrá elegir cuáles son las bandas con los mayores valores de energía de todas en las que ha sido subdividido el sistema previamente.

Esta información es útil para el estudio de vibraciones mecánicas puesto que las bandas con mayores valores de energía representarán los rangos de frecuencia en los que más vibraciones mecánicas se darán, otorgando de esta forma la clave para corregirlos y mejorar el confort y durabilidad del sistema que esté siendo objeto del estudio.

Se detallará la/las banda(s) de energía a la/las que corresponde(n) el/los mayor(es) nivel(es) de energía y el/los valor(es) que toma(n) en un texto que se puede observar en la *Figura 56* continuando con el ejemplo anterior.

Se quieren conocer en el mismo los dos valores más elevados de bandas de energía del sistema.

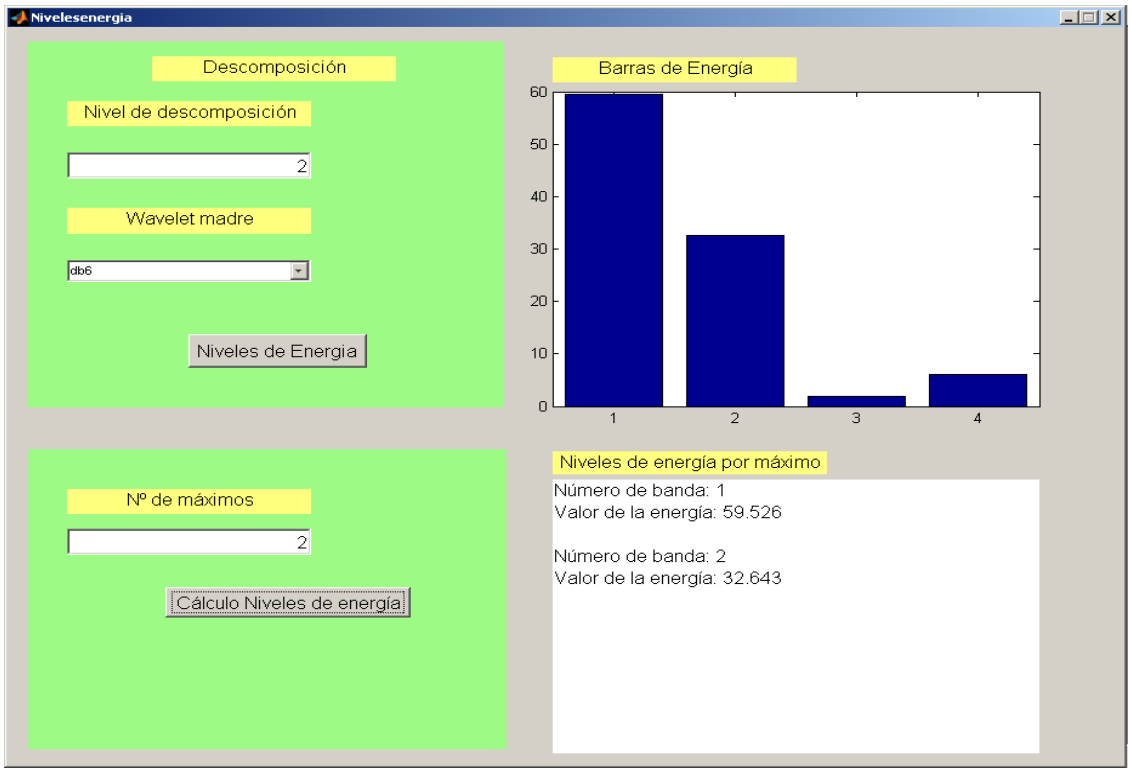


Figura 56: Dos valores máximos de las bandas de energía de una senoide

5.2.8.- Configuración de la tarjeta

El acceso a esta utilidad se lleva a cabo a través de la ruta de acceso correspondiente a *Adquisición de datos/Configuración de la tarjeta.*

La situación inicial al ejecutar la aplicación es la que se presenta en la *Figura 57*.

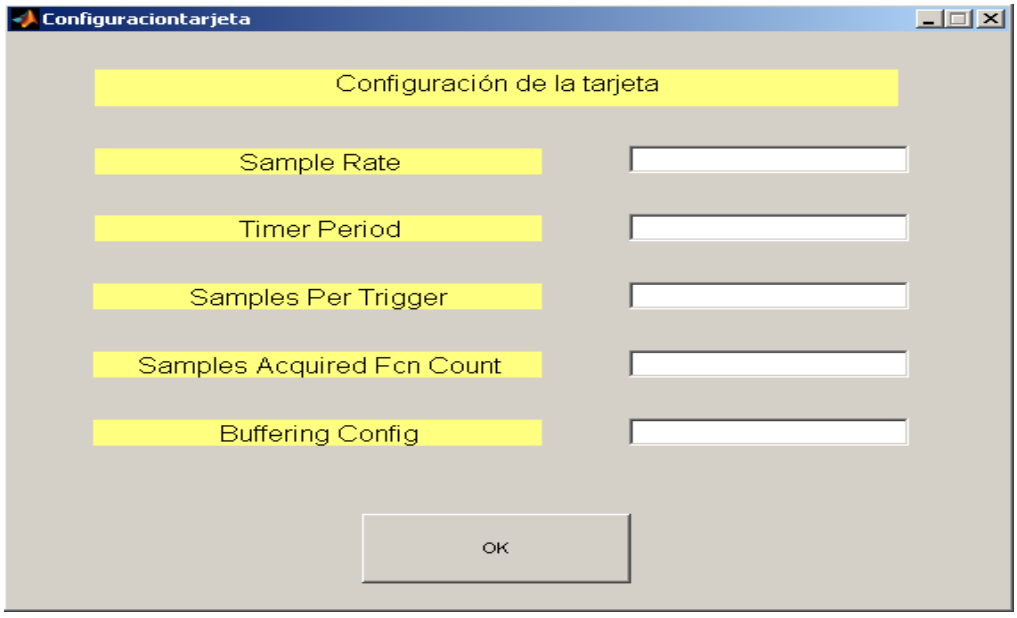


Figura 57: Aplicación para modificar los parámetros de la tarjeta de adquisición de datos

El objetivo al acceder a la ventana *Configuración de la tarjeta* es el de editar las propiedades que se ofrecen en la misma de la tarjeta de adquisición de datos que se conectará al sistema y que recibirá la información del mismo.

Dichas propiedades se enuncian a continuación:

1.- *Sample Rate*: Es la frecuencia de muestreo con la que se quiere trabajar en la toma de datos.

2.- *Timer Period*: Es el periodo de tiempo que se quiere abarcar en el estudio que se esté llevando a cabo.

3.- *Samples Per Trigger*: Es el número de muestras que se toman por disparo de la tarjeta.

4.- *Samples acquired Fcn Count*: Es el contador de muestras tomadas por el sistema.

5.- *Buffering Count*: Es el contador del búfer.

5.3.- Desarrollos con continuidad en el futuro

Se ha decidido añadir una serie de interfaces gráficas en la aplicación Btool que se consideran útiles en cuanto a su posible implementación en un futuro. Se dejan como guía para que se retome el proyecto bajo estas directrices que se consideraban importantes como desarrollo del mismo.

Se trata de aplicaciones que han sido desarrolladas por completo y que esperan códigos que se integren en lo ya construido. Resultará sencillo modificar los archivos y conseguir desarrollar una herramienta de trabajo más potente con la que abarcar un estudio más completo de las señales en general, y del mantenimiento predictivo mediante análisis de vibraciones en particular.

Ambas aplicaciones han sido incluidas dentro del apartado de defectos de la ventana principal, ya que el objetivo de las mismas es un tratamiento de los defectos que aparezcan en los rodamientos.

Se siguieron como patrón de diseño las indicaciones que se recibieron desde el Departamento de Ingeniería Mecánica en cuanto a funcionalidades y elementos que se consideraba oportuno incluir en cada caso. Se han dejado integradas en el proyecto ambas aplicaciones para una más sencilla implementación del código que las haga funcionar correctamente en el futuro.

5.3.1.- Entrenar

Se accede a la misma siguiendo la ruta desde la ventana principal que se detalla: *Defectos/Entrenar*. El aspecto de la aplicación es el detallado en la *Figura 58*.

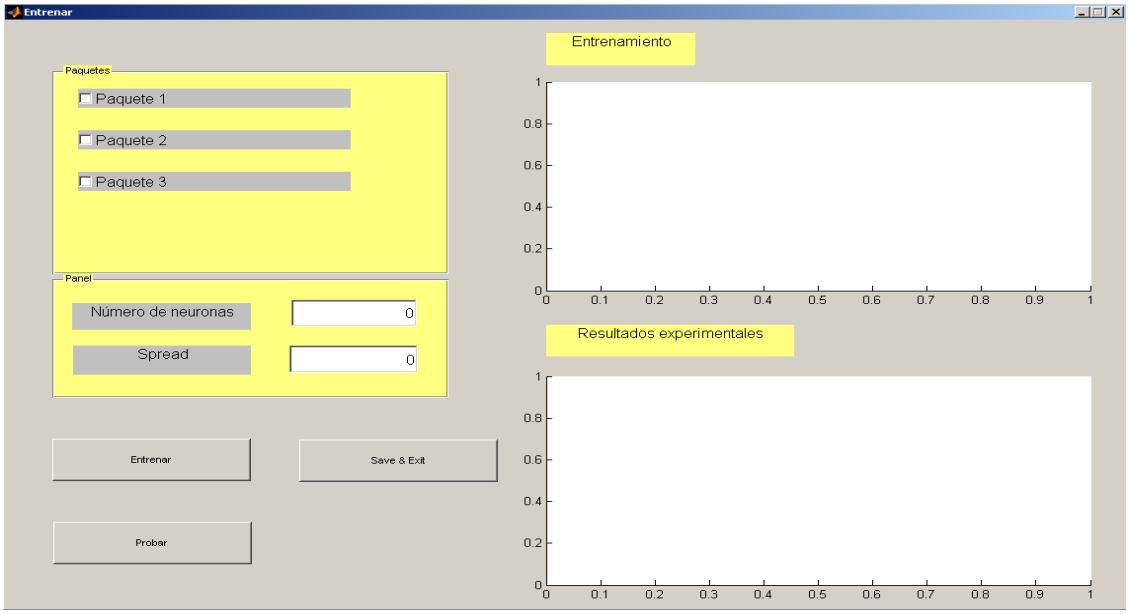


Figura 58: Aspecto de la ventana Entrenar

5.3.2.- Clasificación de defectos

Para acceder a la aplicación que se presenta en este apartado se ha de seguir la ruta a través de *Btool* que se indica, *Defectos/Clasificación*, mostrada en la figura 59.

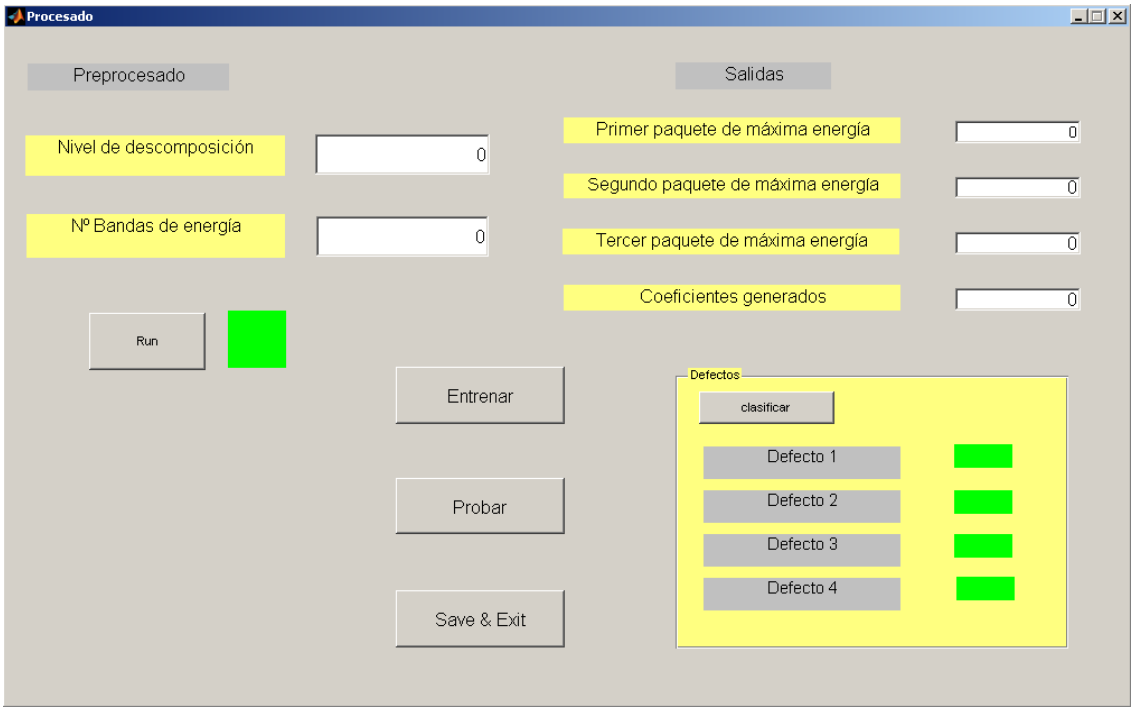


Figura 59: Aspecto de la ventana Clasificación

CAPÍTULO 6: RESULTADOS EXPERIMENTALES

6.- RESULTADOS EXPERIMENTALES

6.1.- Consideraciones iniciales.

Si acaso hubo un momento crucial en el desarrollo del proyecto ese fue el correspondiente al paso de la programación de la interfaz gráfica a la aplicación práctica para la obtención de resultados experimentales.

El sistema en sus primeras fases de adaptación se encontraba vulnerable, debido a que la situación real no iba a situarse en el mismo contexto que a la hora de programar, pudiendo darse imprevistos debidos tanto a agentes externos al sistema como al sistema en sí mismo.

Es por este motivo por el que aún quedaba mucho trabajo por delante ya que, a pesar de que la organización de la programación, el uso de las variables, los tipos de datos y la forma en la que se comunicaban entre sí las distintas aplicaciones no conllevaba ningún error por parte de MATLAB, aún no se habían constatado tangiblemente los resultados experimentales, que resultaban sin duda los más importantes para que el diseño de la aplicación llegara a buen puerto.

Debido a estas razones se decidió llevar a cabo una prueba previa a la aplicación a la medida de vibraciones mecánicas propiamente dicha, en la que se pudiera comprobar que el sistema era capaz de registrar y representar señales entrantes, cualesquiera que fueran de modo satisfactorio.

Muchos eran los aspectos a mejorar, y muchas las líneas de código que se podían depurar para conseguir el funcionamiento más eficiente y efectivo de la aplicación para conseguir resultados que pudieran a posteriori desembocar en una medida adecuada de las señales a las que iba a ser expuesto el sistema.

Se decidió pues contar con una serie de fuentes y generadores que serían conectados a la tarjeta de adquisición de datos, diferenciando ostensiblemente unos de otros para poder así identificar cada uno de ellos claramente.

Para ello se dispusieron distintos niveles de tensión en las fuentes de continua, y distintas frecuencias, amplitudes y formas de onda en los generadores de funciones.

No se seleccionaron éstas entradas al sistema aleatoriamente, sino por el motivo de que, al conectar entradas cuyos valores se determinan y por tanto, se conocen previamente a la conexión de las mismas al sistema, el resultado esperado a la hora de representar las señales era inequívoco; pudiendo así constatar inmediatamente si la herramienta estaba tomando datos lógicos o no de las fuentes de información a las que estaba conectada.

Fue una fase exigente en cuanto a detección de fallos y programación, en la que se buscó no solamente el funcionamiento básico de la aplicación, sino también ir mejorando paulatinamente el código para desembocar en el mejor resultado posible.

A continuación se describe el proceso llevado a cabo y los resultados que se obtuvieron de dichas pruebas previas a la conexión con el sistema en el que se centra el diseño de la aplicación que ocupa este proyecto

6.2.- Pruebas de validación de la herramienta.

La prueba consistió en conectar los siguientes generadores a cinco de los ocho canales de los que dispone la aplicación *Btool*, que eran los siguientes (figura60):

- 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.

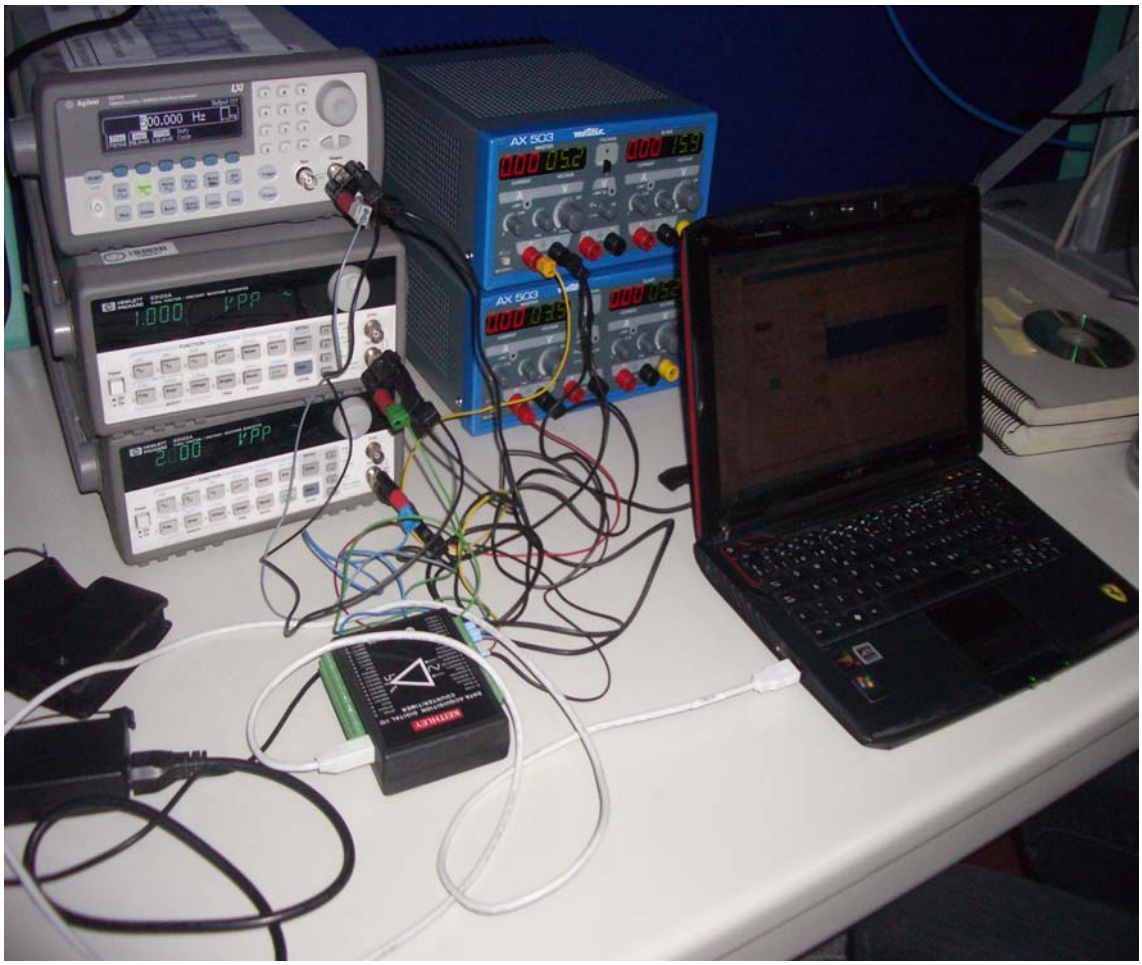


Figura 60: Entorno de pruebas de puesta a punto de la herramienta.

En la figura se puede apreciar el conexionado que se llevó a cabo para la toma de datos, apreciándose cómo se conectaba cada una de las fuentes a las entradas analógicas de la tarjeta de adquisición de datos del modo que se presenta en la *Figura 61*.

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

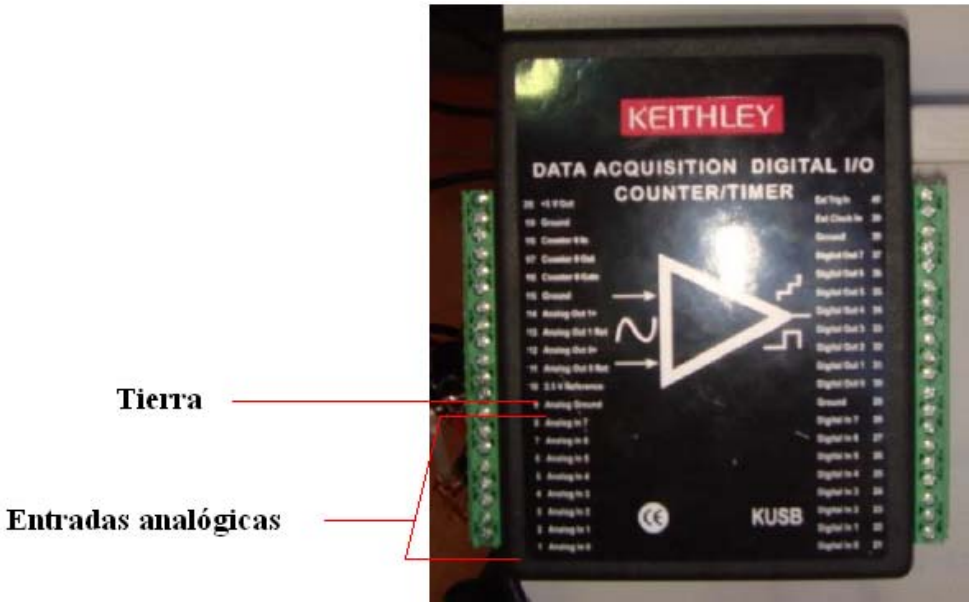


Figura 61: Entradas analógicas y tierra de la tarjeta de adquisición de datos

A continuación se presenta el resultado que dio por pantalla las pruebas realizadas con el montaje anteriormente explicado y expuesto.

En primer lugar se quiso probar la aplicación Select Channel, debido a que al medir un canal exclusivamente a seleccionar entre los ocho posibles, resultaba más sencillo poder identificar y depurar errores canal por canal. Tras haber solucionado los errores que se fueron presentando, tanto de hardware como de software se tuvo un resultado satisfactorio.

Se sometió a la aplicación a la toma de varios tipos de señales para comprobar el correcto funcionamiento de la aplicación. En la *Figura 62* se le introduce una señal sinusoidal y en la *Figura 63* se toma como fuente de datos una señal cuadrada.

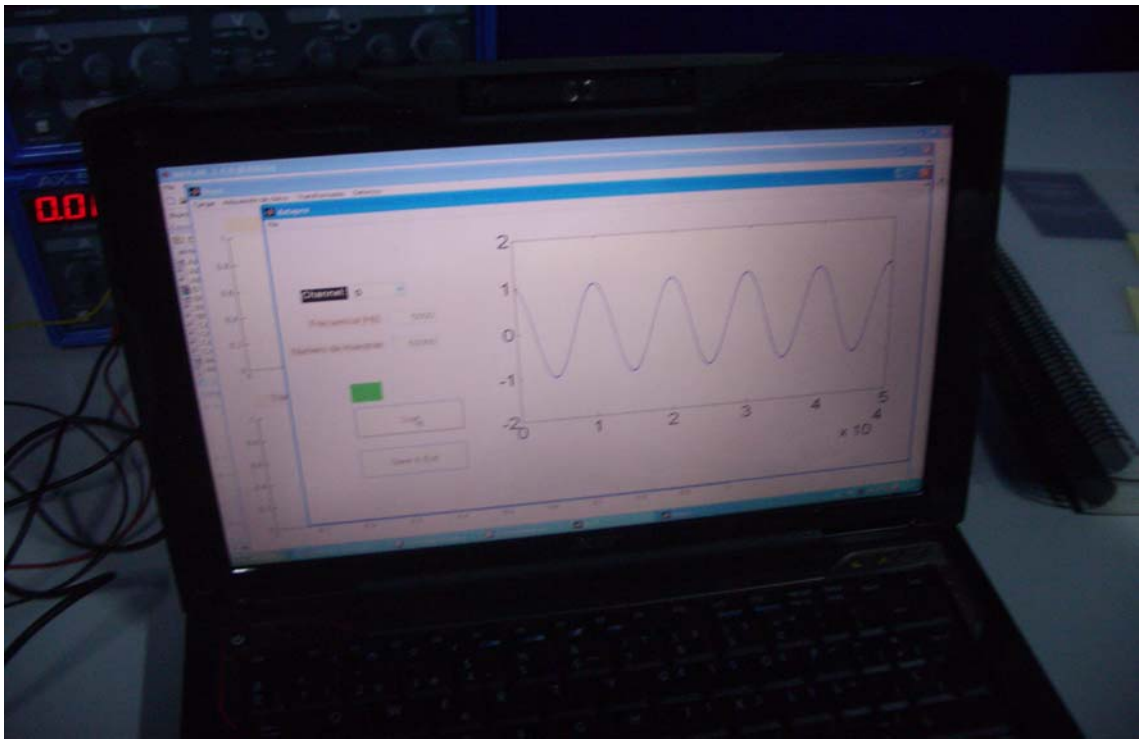


Figura 62: Prueba de la aplicación Select Channel con señal sinusoidal

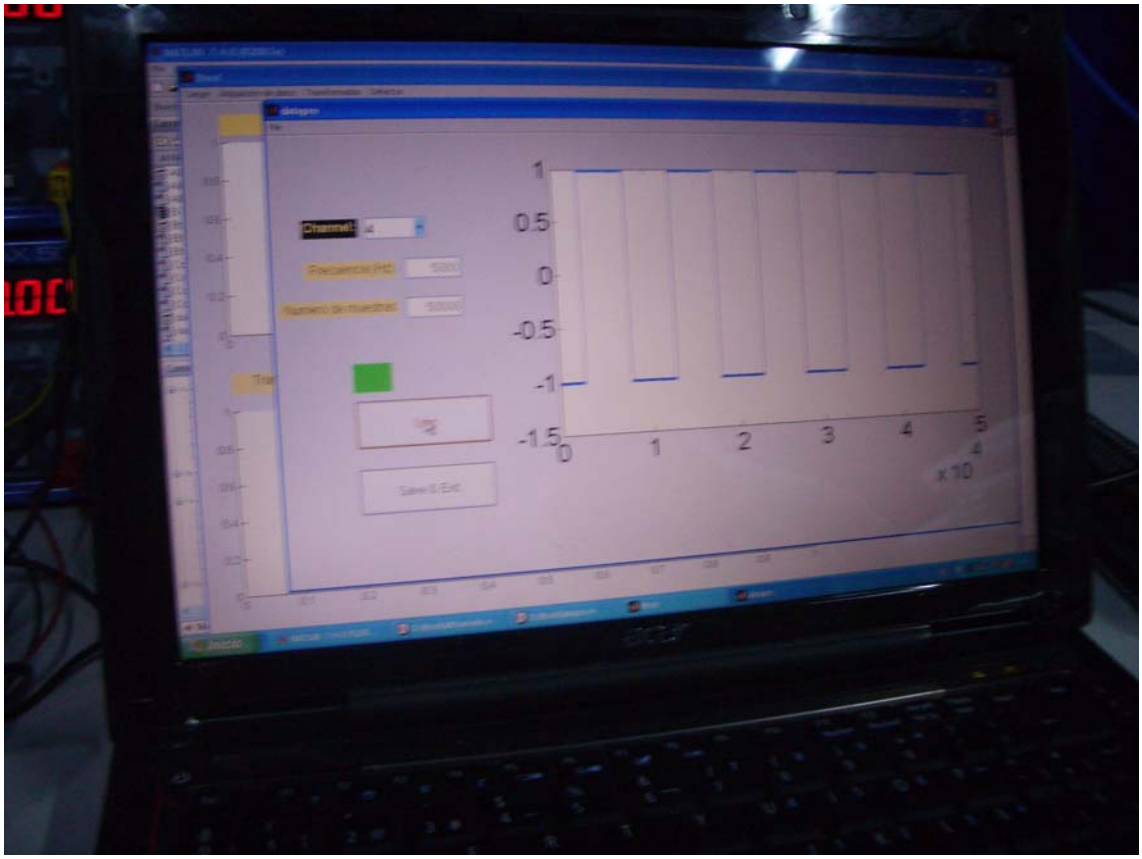


Figura 63: Prueba de la aplicación Select Channel con señal cuadrada

El objetivo era asegurarse de que el sistema funcionaba adecuadamente y por este motivo se le sometió a diversas pruebas, como por ejemplo comprobar que la comunicación entre la aplicación Select Channel y la ventana principal de la aplicación era correcta.

Una vez se lograron las premisas impuestas se pasó a analizar la herramienta *All Channels.*

Es la situación que mayor exigencia requiere a la aplicación *Btool,* y tras conseguir que funcionase correctamente los resultados obtenidos por la prueba fueron los que se muestran en la *Figura 64*.

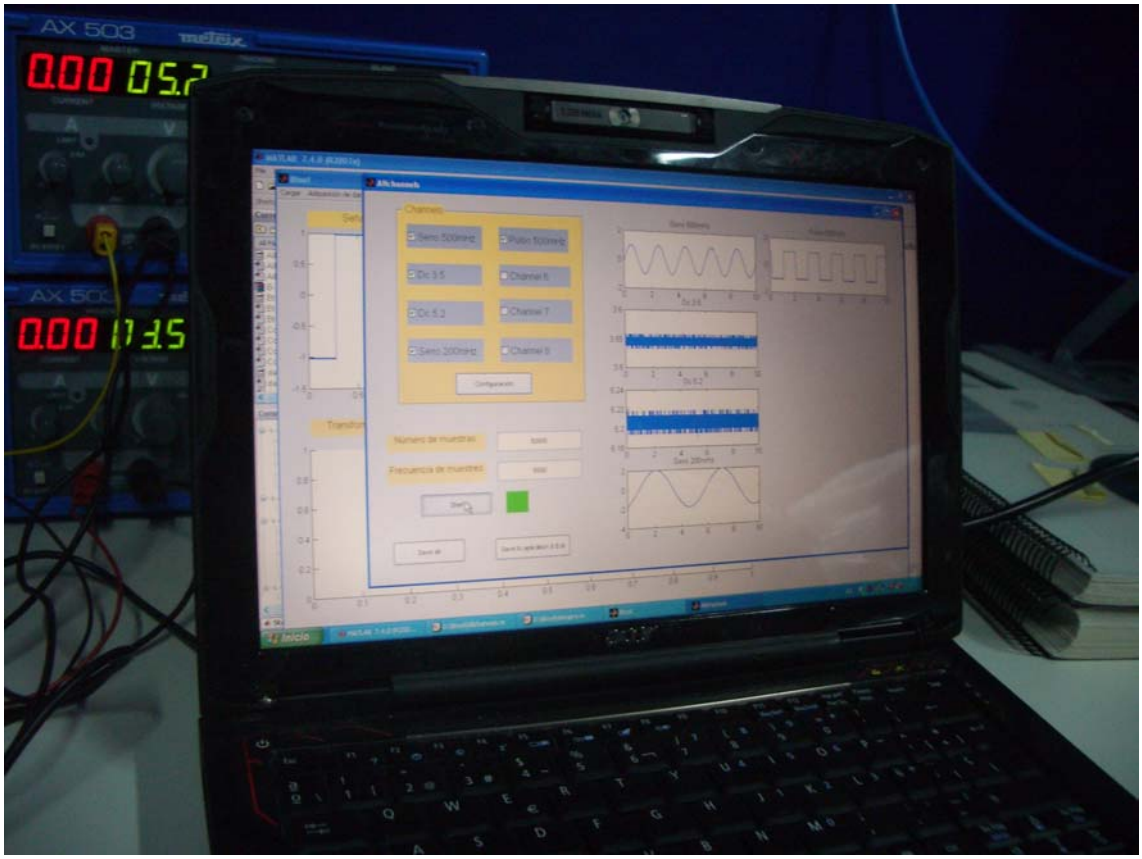


Figura 64: Prueba de la aplicación All Channels con las cinco señales conectadas

Se pueden observar los cinco canales que se indican, los cuales en el momento de la captura de pantalla se encontraban en las siguientes condiciones:

- Canal 1: Representa una señal sinusoidal de 500Hz de frecuencia y $1V_{pico}$.
- Canal 2: Representa una señal de DC con valor 3,5V.
- Canal 3: Representa una señal de DC con valor 5,2V.
- Canal 4: Representa una señal sinusoidal de 200Hz de frecuencia y $2V_{pico}$.
- Canal 5: Representa una señal cuadrada de 500Hz de frecuencia y $1V_{pico}$.

Sorprendió gratamente la exactitud con la que se obtenían los datos, hecho que se manifiesta claramente en el caso de las fuentes de DC.

Se puede ver cómo la herramienta *All Channels* representa las pequeñas variaciones de tensión que siempre ofrece una fuente de DC, oscilando los valores, por ejemplo en el caso del canal 3, entre 5,18V y 5,22V.

Una vez alcanzados los resultados esperados el sistema estaba preparado para ser probado en el entorno para el que fue expresamente diseñado.

6.3.- Toma de datos de análisis de vibraciones

Una vez que se ha preparado la aplicación Btool, se ha procedido a su prueba en el entorno real de trabajo, es decir, en el Machine Fault simulator Lite (MFS Lite) que se encuentra en el laboratorio del Departamento de Ingeniería Mecánica.

Se han conectado las señales que proceden del acelerómetro tras pasar por los amplificadores a la tarjeta KUSB 300 de Keithley, unida a uno de los puertos USB del ordenador.

En algunos casos se han considerado únicamente los ejes X e Y del acelerómetro, pues son los que proporcionan la información necesaria para el estudio a realizar.

Para la realización de las pruebas, se han considerado el análisis de un rodamiento con defecto en pista exterior, haciendo pruebas a 10, 20 y 40 Hz. Tras la toma de datos, se ha procedido a su análisis espectral. Para los tres casos se muestran las gráficas tomadas, la gráfica tras el análisis espectral y las conclusiones.

6.3.1 Rodamiento con defecto en pista exterior a 10 Hz

En la figura 65 se muestran las lecturas realizadas sobre el rodamiento con defecto en pista a 10 Hz.

Los parámetros de la toma de datos son:

- Número de muestras: 5000
- Frecuencia de muestreo: 5000Hz

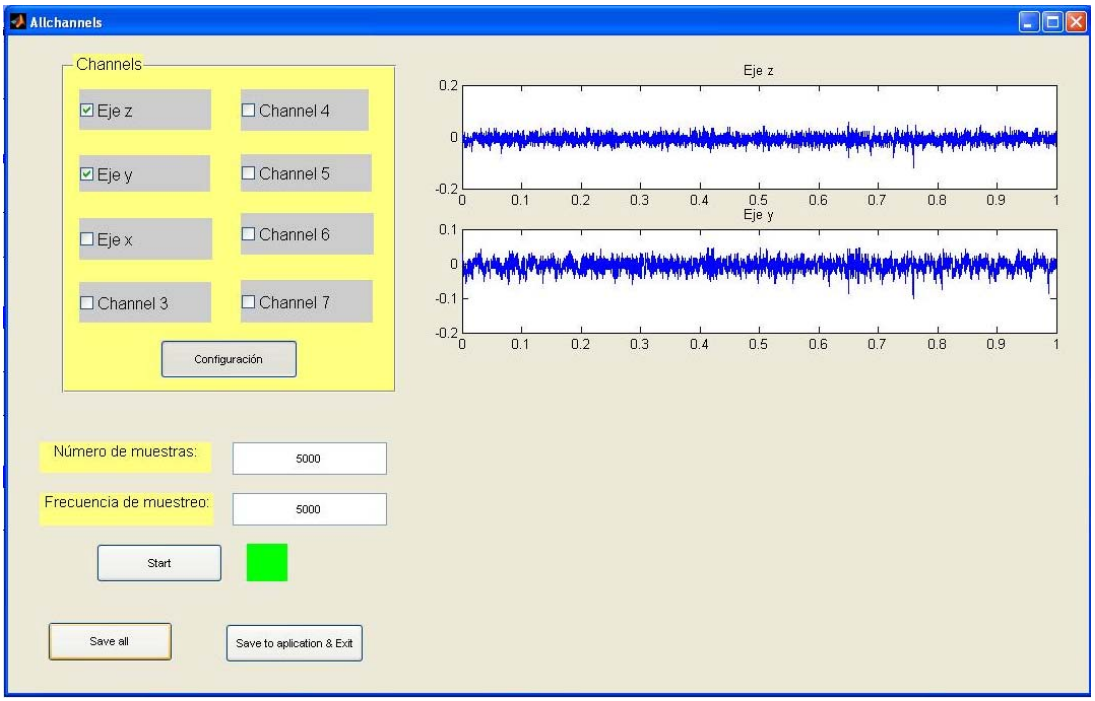


Figura 65: Rodamiento con defecto en pista exterior a 10 Hz: datos tomados

En la figura 66 se muestra el resultado después de aplicar el análisis espectral, donde cabe observar la frecuencia característica del rodamiento con defecto en la pista externa, que por medios teóricos se ha calculado de un valor de 30.52 Hz.

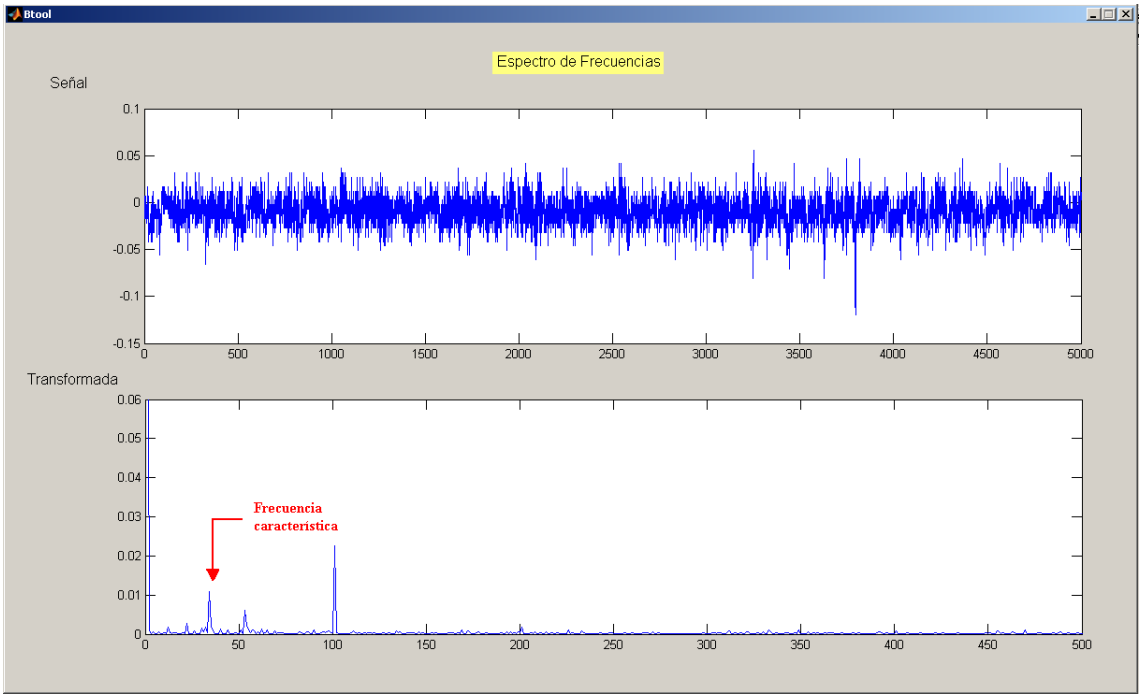


Figura 66: Rodamiento con defecto en pista exterior a 10 Hz: espectro de frecuencias. Eje y

Ahora se calcula la descomposición de Wavelets, tomando los siguientes parámetros:

- Nivel de descomposición: 2
- Wavelet madre: Daubechies 6
- Número de máximos: 2

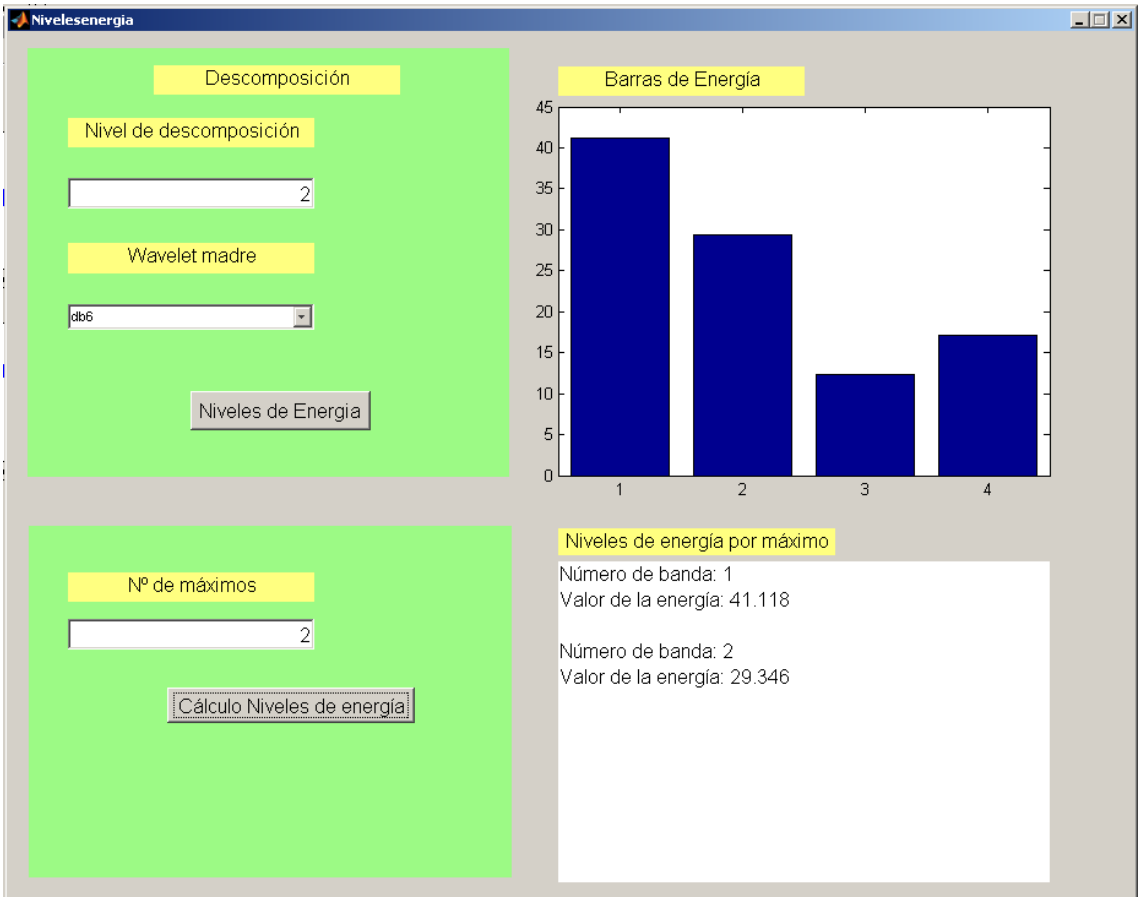


Figura 67:Rodamiento con defecto en pista exterior a 10 Hz:Descomposición en Wavelets Eje y

Se ve que la banda de energía con mayor valor corresponde con el rango espectral en el que está contenida la frecuencia de defecto.

6.3.2 Rodamiento con defecto en pista exterior a 20 Hz

En la figura 67 se muestran las lecturas realizadas sobre el rodamiento con defecto en pista a 20 Hz.

Los parámetros de la toma de datos son:

- Número de muestras: 5000
- Frecuencia de muestreo: 5000Hz

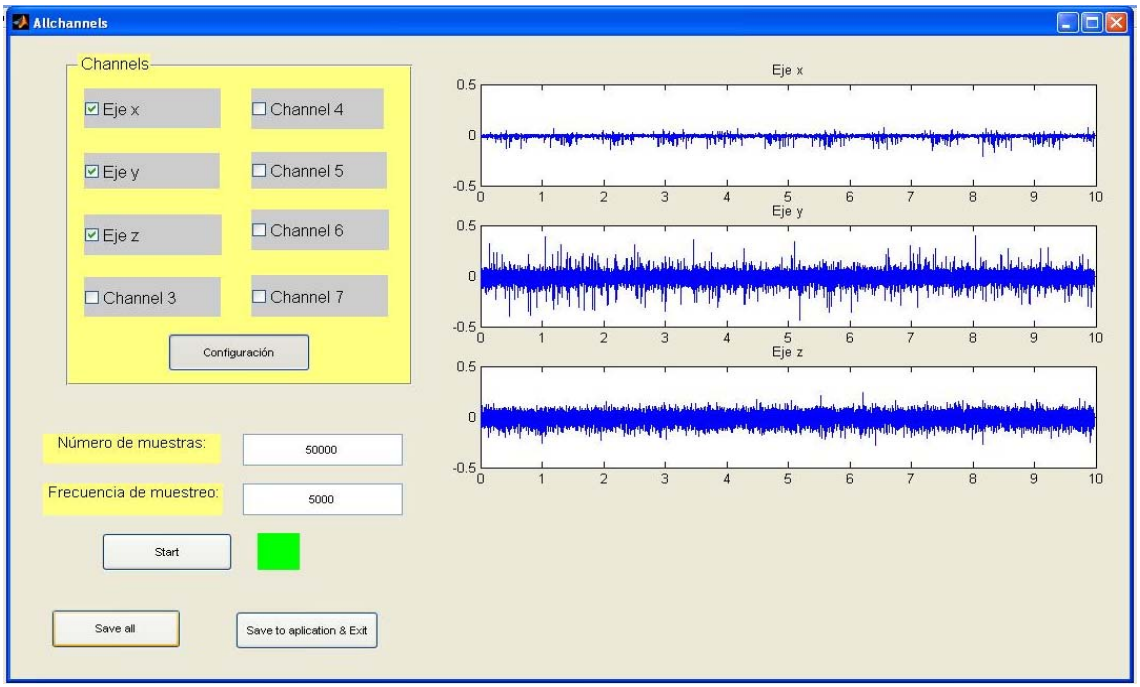


Figura 68: Rodamiento con defecto en pista exterior a 20 Hz: datos tomados

En la figura 68 se muestra el resultado después de aplicar el análisis espectral, donde cabe observar que el número de perturbaciones en el mismo ha aumentado considerablemente respecto a los valores tomados a 10 Hz.

La frecuencia de defecto teórica resulta ser de 48.04 Hz.

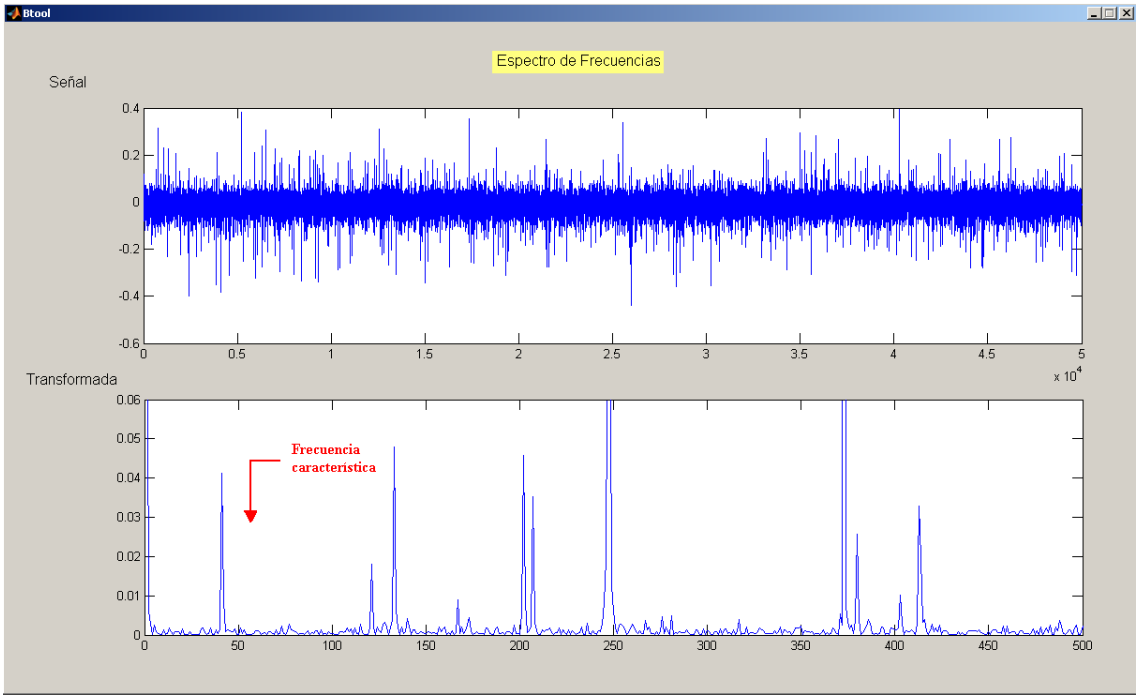


Figura 69: Rodamiento con defecto en pista exterior a 20 Hz: análisis espectral. Eje y.

Ahora se calcula la descomposición de Wavelets, tomando los siguientes parámetros:

- Nivel de descomposición: 2
- Wavelet madre: Daubechies 6
- Número de máximos: 2

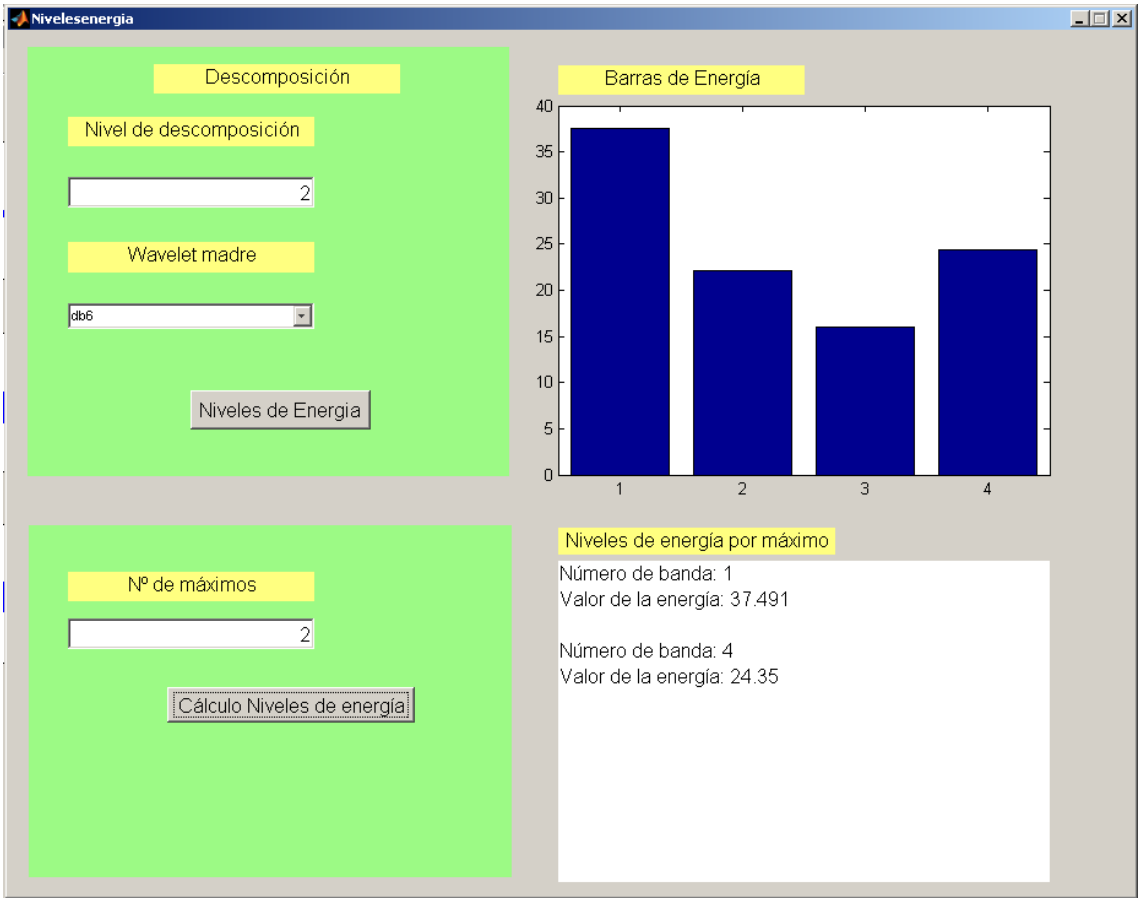


Figura 70:Rodamiento con defecto en pista exterior a 20 Hz:Descomposición en Wavelets Eje y

Sigue siendo el valor de la banda de energía mayor el que se correponde con el rango espectral correspondiente a la frecuencia de defecto.

6.3.3 Rodamiento con defecto en pista exterior a 40 Hz

En la figura 69 se muestran las lecturas realizadas sobre el rodamiento con defecto en pista exterior a 40 Hz.

Los parámetros de la toma de datos son:

- Número de muestras: 5000
- Frecuencia de muestreo: 5000Hz

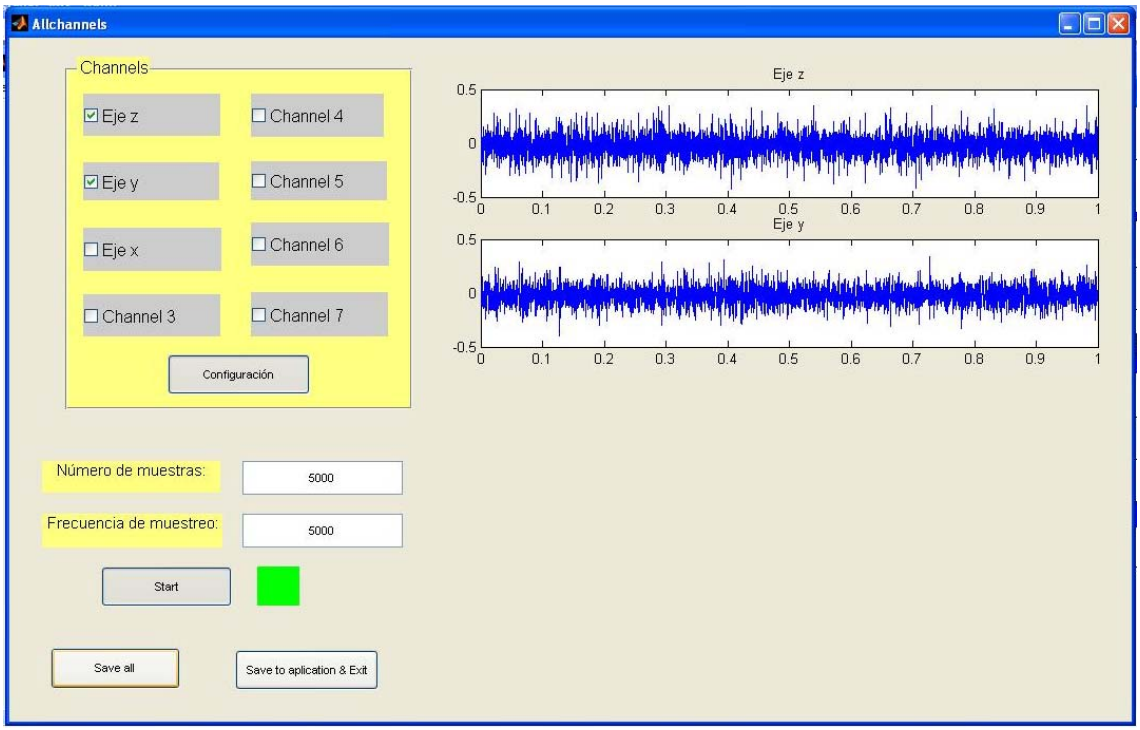


Figura 71: Rodamiento con defecto en pista exterior a 40 Hz: datos tomados

En la figura 70 se muestra el resultado después de aplicar el análisis espectral, donde cabe observar que el contenido de frecuencias del mismo es tan elevado que es complicado discernir la frecuencia de defecto.

Por métodos teóricos se llega a que dicha frecuencia se sitúa en torno a los 121.64 Hz

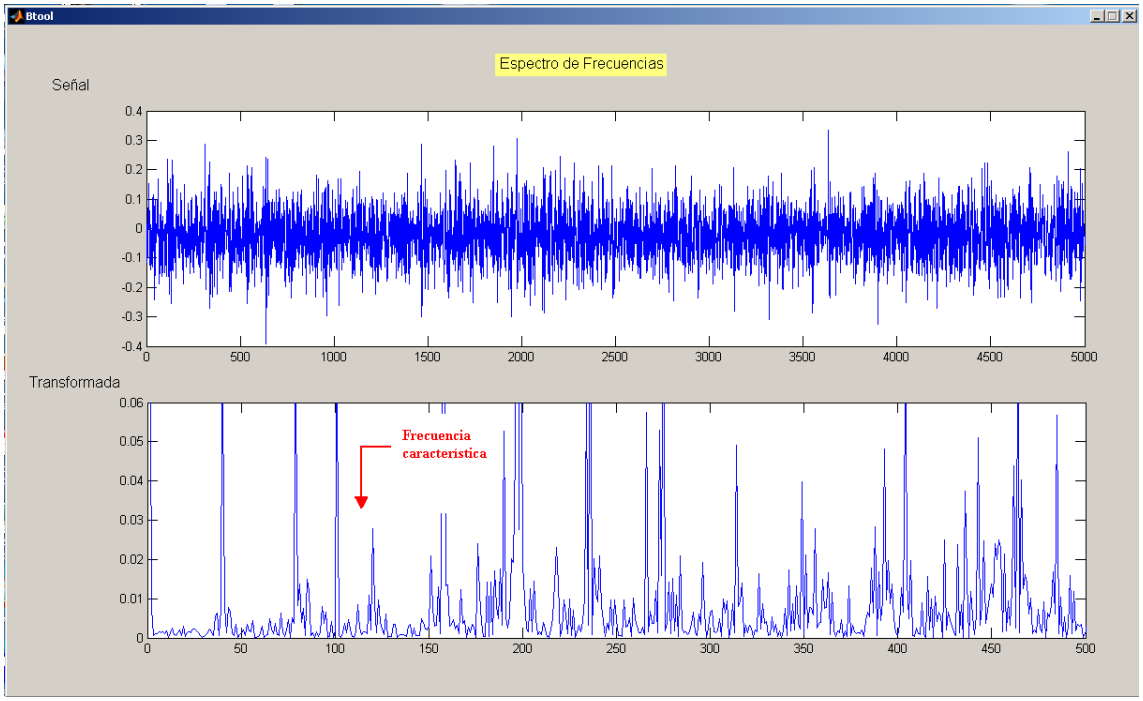


Figura 72: Rodamiento con defecto en pista exterior a 40 Hz: análisis espectral. Eje y

Ahora se calcula la descomposición de Wavelets, tomando los siguientes parámetros:

- Nivel de descomposición: 2
- Wavelet madre: Daubechies 6
- Número de máximos: 2

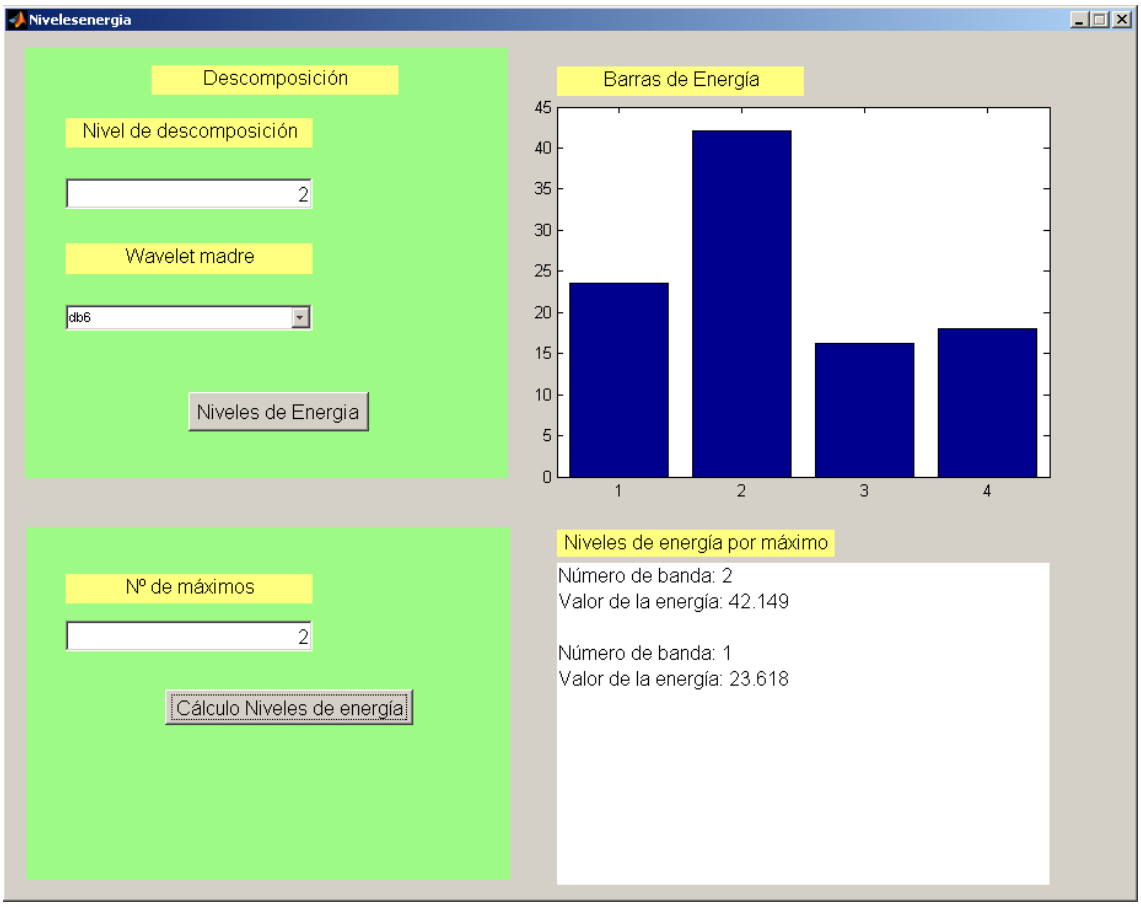


Figura 73:Rodamiento con defecto en pista exterior a 40 Hz:Descomposición en Wavelets Eje y

El análisis por Wavelet constata lo comentado acerca de la figura 72. El contenido en perturbaciones en frecuencia mayor no se corresponde con el rango en el que se encuentra la frecuencia de fallo del rodamiento con defecto en pista externa.

CAPÍTULO 7: CONCLUSIONES Y TRABAJOS FUTUROS

7.- CONCLUSIONES Y TRABAJOS FUTUROS

7.1 Conclusiones

Como conclusión, se ha desarrollado y probado una herramienta con MATLAB que permite la adquisición de señales mediante el manejo de una tarjeta de adquisición de datos destinada al estudio de defectos en rodamientos, sustituyendo de este modo al software que se venía empleando antes de la aparición de la aplicación (software de la tarjeta de adquisición Keithley), permitiendo mayor versatilidad y sencillez de manejo, tanto como para los investigadores y profesores como para los alumnos que emplee esta herramienta para hipotéticas prácticas de laboratorio.

Se ha desarrollado la interfaz gráfica, resuelto los problemas de conexión con el hardware y probado la robustez del sistema, con lo que se entrega una aplicación testada que ofrece grandes garantías de cara a un funcionamiento sin problemas.

Se ha particularizado el uso de la aplicación al estudio de los defectos de los rodamientos, destinándola a realizar diferentes análisis de las señales obtenidas de los acelerómetros.

El sistema ha sido probado en un banco real de rodamientos, validándose así su utilidad bajo otras condiciones de trabajo diferentes a las que suponen el caso específico.

7.2 Trabajos futuros

El proyecto ha desembocado en una aplicación flexible y dinámica, que amplía mucho las expectativas en cuanto a tratamiento de datos de vibraciones mecánicas respecto a sus antecedentes.

Se han implementado en una única aplicación la mayoría de los aspectos a considerar dentro del estudio integral para desembocar en una estrategia más satisfactoria del mantenimiento predictivo.

Sin embargo, aún quedan bastantes aspectos que mejorar en el futuro, para poder así conseguir una aplicación que ofrezca un abanico aún más extenso de herramientas, a la vez que se siguen implementando con el objetivo de mejorar en lo ya diseñado.

Una opción que se postula como más que interesante es la de conseguir representar los datos en las gráficas en tiempo real, creando un modo de trabajo donde la aplicación funcione como si de un osciloscopio se tratara. Se representarían las señales en tiempo real y se podría observar la evolución de las mismas en todo momento.

Se podría asimismo seguir añadiendo distintos métodos de procesamiento de señales en el apartado correspondiente a *Transformadas* de una forma extremadamente sencilla.

Basta con anexar el código al presente proyecto y se podrá comenzar a trabajar inmediatamente con las mejoras sin ninguna clase de problema.

Puede tratarse tanto de métodos presentes en la actualidad como de nuevas ideas, de modo que la aplicación se va a presentar como flexible a los cambios que puedan tener lugar en el estudio de señales vibratorias; evitando quedarse obsoleta en breve espacio de tiempo, lo cual es importante, pues es un campo que se encuentra actualmente en continua innovación y cambio.

Otra posible mejora puede ser la de incorporar la opción de realizar un *zoom* localizado en alguna zona en concreto de las gráficas que, por su naturaleza o especial relevancia en el análisis llevado a cabo, se quiera resaltar para un estudio más pormenorizado.

Como trabajo que se ha dejado hecho para que futuros proyectos lo lleven a cabo se encuentran los apartados de defectos de los rodamientos; habiéndose creado interfaces gráficas en las que se podrán integrar los códigos correspondientes para un funcionamiento satisfactorio de manera cómoda.

Sin duda es un campo que se está investigando en la actualidad, y que sin duda se seguirá tratando en el futuro, dados los resultados satisfactorios que está arrojando; es por ello por lo que se ha diseñado una herramienta completa, pero siempre abierta al cambio y la incorporación de funcionalidades, dotándola así de una flexibilidad que se antojaba necesaria para obtener resultados satisfactorios.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

[1] RENOVETEC, "Ingeniería de mantenimiento". Colección mantenimiento industrial. disponible en Internet en www.renovetec.com

[2] DUQUE PÉREZ, O., "Técnicas de mantenimiento preventivo". Ed. Abecedario, (2009).

[3] SKF. "SKF Maintenance Products", SKF ©, (2008)

[4] WHITE, G. "Introducción a la vibración de máquinas". Part number 8569, version 1.75, DLI Engineering, (1995).

[5] 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).

[6] BRAUN, S. "Analysis of Roller/Ball Bearing Vibrations". ASME Publication (1979).

[7] 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).

[8] MEIROVITCH, L. "Elements of vibration analysis". Second edition. Mc Graw-Hill, (1986).

[9] BARRAGÁN GUERRERO, D. "Manual de interfaz gráfica de usuario en Matlab". Parte 1. www.matpic.com (2005).

[10] THE MATHWORKS INC. "GUI Building Tutorial" (2005). Disponible en: http://www.mathworks.es/academia/student_center/tutorials/launchpad.html.

[11] CHANG, R. "Química", (2007).

[12] LARA, J. "Nuevas metodologías no invasivas de diagnosis de defectos incipientes en rodamientos de bola". Tésis doctoral. Universidad Carlos III. Madrid. Departamento de Ingeniería Mecánica, (2007).

[13] GONZÁLEZ RODRÍGUEZ, J., "Aplicación del método shok pulse para detección de defectos en rodamientos". Proyecto fin de Carrera. Ingeniería Industrial. Departamento de Ingeniería Mecánica, (2009).

[14] POLIKAR, R., "The Wavelet tutorial". Rowan University. College of Engineering. (1999). Tutorial también disponible en Internet en el link: http://users.rowan.edu/polikar/WAVELETS/WTutorial.html

[15] KAPLAN, I., "Wavelets and signal processing" (2003). Tutorial disponible en Internet en el link: http://www.bearcave.com/misl/misl_tech/wavelets/

[16] MARTÍN MARTÍNEZ, J., "Diccionario de la informática". Editorial Ra-Ma. (2001).