Universidad de Valladolid

E.T.S. DE INGENIERÍA INFORMÁTICA

Máster en Ingeniería Informática

Implementación software de un amplificador Lock-in

Alumno: Eduardo Rodriguez Gutiez

Universidad de Valladolid

E.T.S. DE INGENIERÍA INFORMÁTICA

Máster en Ingeniería Informática

Implementación software de un amplificador Lock-in

Alumno: Eduardo Rodriguez Gutiez

Tutor: Helena Castán Lanaspa
Salvador Dueñas Carazo

Resumen

Cualquier ingeniero conoce el significado de ingeniería, que, en caso de que el lector lo desconozca, consiste en llevar a cabo unos objetivos contando con ciertas restricciones de tiempo y costes, entre otros. Como resulta lógico pensar, todo objetivo y restricción implica unos márgenes, o, dicho de otro modo, unas tolerancias. Es totalmente imposible realizar cualquier trabajo con precisión absoluta, ya sea una pieza con un grosor de exactamente 2 milímetros, obtener un compuesto totalmente puro, lanzar un trabajo a las 12 horas de un día, o sintetizar o medir una señal perfectamente, sin interferencia alguna de fuentes de ruido externas.

A medida que la ciencia y la tecnología avanzan, se va imponiendo el uso de tolerancias cada vez más pequeñas, lo que permite que otras disciplinas, a su vez, mejoren. Gracias a una mayor precisión en la mecánica, ha sido posible la creación de robots para telemedicina y radioterapia, la obtención de compuestos exentos de impurezas ha permitido mejores análisis de sus propiedades, lo que a su vez ha permitido crear y pronosticar la existencia de nuevos materiales como plásticos y espumas. El control del tiempo es un factor esencial en todo el sector de las TIC, permitiendo el desarrollo de la carrera espacial y de la geolocalización. Por último la mejora en la síntesis, adquisición y tratamiento de las señales ha hecho posible desde la conexión de millones de teléfonos móviles a la red hasta la existencia del instrumental de análisis que utilizan químicos (como son los espectrofotómetros y espectrómetros en general) y físicos (estabilización de frecuencia en láseres), pasando por las altas velocidades de transmisión de las que hoy disfruta cualquier hogar medio.

El fin de este Proyecto Fin de Máster, que lleva por título “Implementación software de un amplificador Lock-in”, está directamente relacionado con éste último. Como ya se ha comentado, en el mundo físico resulta casi imposible que una señal esté totalmente libre de ruido, ya que afectan, entre otros, del movimiento de los portadores de carga a través de los materiales y la temperatura. Eso significa que las resistencias presentes en cualquier dispositivo electrónico, al pasar la corriente a través suyo, generan un ruido que, si bien puede reducirse, no es posible eliminar. En otros casos, resulta necesario obtener señales provenientes de variaciones muy débiles en cierta magnitud física o que están afectados por ruido varios órdenes de magnitud superiores a la propia señal (lo que técnicamente se conoce como una relación señal ruido muy baja). La idea detrás de un amplificador Lock-in es que, si conocemos la frecuencia de la señal de interés, podemos recuperar la fase y magnitud de la onda, así como la componente en fase y en cuadratura de la misma. Esto resulta especialmente útil para el análisis de la conductancia y la capacitancia de un circuito, que, además de su aplicación directa en la determinación y caracterización de forma precisa de componentes electrónicos y circuitos, también tiene aplicación en básculas de precisión. Otros usos de un Lock-in incluyen voltímetros vectoriales, medición de ruido, medición de desfase, etc.

Si ya existe entonces una serie de diseños que cumplen con este objetivo, ¿por qué diseñar algo que ya está realizado? A pesar de lo que pueda parecer, la gran mayoría de los sistemas Lock-in existentes están basados en procesadores digitales de señal (DSP), o microcontroladores, o bien están implementados en entornos o lenguajes de programación, como LabView, que no admiten, a priori, que el diseño pueda expandirse y extenderse rápidamente, y funcionar sobre múltiples plataformas con pocos cambios en el código. El objetivo de este proyecto es el de diseñar e implementar un amplificador Lock-in de forma completamente modular, utilizando un lenguaje estándar, tal como C, C++ o sus derivados. De esta forma resulta posible que la aplicación funcione tanto en un PC bajo sistema operativo Windows o Linux, como en un DSP o microcontrolador de última generación. Por ejemplo, algunos chips de la serie PIC32MZ de Microchip disponen de 2MB de Flash, funcionan a una velocidad de hasta 200MHz y tienen disponible un compilador orientado a objetos, concretamente en C++. En este caso, se utilizará el lenguaje C# que, si bien no es directamente aplicable a este tipo de dispositivos, si que puede portarse fácilmente a C++ respetando la estructura de clases. La elección del lenguaje permite, sin embargo, que el sistema pueda funcionar sobre un PC con Windows, algún sabor de Linux u OS X, en estos últimos con ayuda del entorno Mono.

Especificando un poco más, la finalidad de este proyecto es la de obtener un software que funcione sobre el sistema operativo Windows, concretamente 8.1, utilizando el lenguaje de programación C#, con Visual Studio 2012 como entorno de desarrollo, que sea capaz de realizar las funciones de un amplificador Lock-in, tanto mediante simulación exclusivamente, como utilizando un generador de funciones y un osciloscopio conectados al ordenador. El software deberá ser capaz de sintetizar formas de onda definidas por el usuario y enviárselas al generador de ondas para que, a través de su conversor D/A los datos pasen a valores de voltaje o de intensidad utilizando un amplificador de transconductancia, que atravesará un dispositivo bajo prueba (D.U.T., *device under test*). El osciloscopio se encargará de recoger los valores de señal y pasarlos digital mediante su conversor A/D. El módulo software procesará las señales de entrada y de salida para que, mediante el componente de Lock-in, se puedan extraer las componentes de fase y cuadratura, y con ello, la capacitancia y conductancia del D.U.T. En este caso, se utiliza el osciloscopio DSO-X 3104A de Agilent (ahora Keysight) como generador de señales y como osciloscopio.

La idea de realizar un amplificador Lock-in para PC, funcionando tanto en modo simulación exclusivamente, como utilizando el osciloscopio, nos fue sugerida a mi compañero Andrés Escobar Cotán como a mí por parte de nuestros tutores Salvador Dueñas Carazo y Helena Castán Lanaspa, a la que se le han ido añadiendo pequeñas modificaciones y mejoras, algunas para poder solventar los problemas que hemos ido encontrando en su desarrollo. Esta ha sido una ocasión única en la que se nos ha presentado la oportunidad de aprender sobre un tema que desconocíamos, con lo cual optamos por una planificación basada en el modelo iterativo, para ir realizando revisiones a medida que íbamos profundizando en un tema tan extenso como es el de procesado de señales. Debido a ello, parte de la funcionalidad requerida no ha podido ser implementada, para no alargar excesivamente el proyecto. Finalmente, el resultado será dividido en dos defensas diferentes; mi compañero Andrés se centrará en la comunicación con el osciloscopio, y yo expondré, a lo largo de esta memoria, la simulación del sistema físico del que forma parte el amplificador.

Palabras clave Lock-in, Ruido, Señal, Amplificación, Onda, Filtro, Procesado Digital de Señales, Transformada de Fourier, Osciloscopio, Instrumentación, GPIB, C#, Programación orientada a objetos

A mis abuelos Carmen y Rufino,
a mis padres Julio y Mª del Carmen,
a mis tíos y primos.

Agradecimientos

Existe un dicho en inglés que reza "No man is an island", que pone de relieve las relaciones que se crean de forma natural entre las personas. De la misma forma, puede decirse sin lugar a dudas que cualquier clase de trabajo, sobre todo en el campo de la ciencia y de la ingeniería, no es un ente aislado, sino que no hubiera sido posible sin las bases que algunos pioneros crearon antes que nosotros. Es justo reconocer que, además de estas bases, un proyecto también resultaría imposible de realizar de no ser por los cientos de pequeños comentarios, aportes y mejoras realizadas por personas más cercanas con las que estamos en deuda:

- *A María del Carmen Gutiez De Hoyos y Julio Rodriguez Rodriguez, mis padres, por su incesante apoyo a lo largo de toda mi vida.*
- *A Héctor García García, que a lo largo del proyecto nos ha proporcionado múltiples referencias, y ha contribuido con algunas soluciones en la parte matemática.*
- *A Jesús Arias Álvarez, por sus detalladas y amenas explicaciones sobre el fundamento matemático y electrónico que existe detrás los amplificadores y por proveer varios ejemplos de aplicaciones de los Lock-in.*
- *A Manuel Ángel González Delgado, por sus comentarios y explicaciones acerca de la transformada de Fourier.*
- *A Sergio Ferrero Martín y Héctor Barbero, por las aportaciones sobre la aplicación de los Lock-in a diversos campos.*
- *A mi compañero Andrés Escobar Cotán, por todas las horas que hemos pasado en común desarrollando este proyecto en el laboratorio.*
- *A nuestros tutores, Helena Castán Lanaspa y Salvador Dueñas Carazo, por su paciencia y constancia en las tutorías, y sobre todo, por proponernos y animarnos a la realización de este proyecto.*
- *A Isabel Carriazo Farré, por cederme la plantilla de LaTeX que utilicé en mi PFC y vuelvo a utilizar en este TFM, y a Guillermo Rodríguez Cano, por crearla.*
- *A todas aquellas personas que también han realizado colaboraciones y que, ya que por desgracia la memoria humana no es pefecta, no aparecen en esta página.*

A todos vosotros, Gracias.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

"If opportunity doesn't knock, build a door"

Milton Berle

1.1. Descipción y ámbito del proyecto

En este apartado se exponen las instituciones, personas, procesos y productos afectados por el desarrollo del proyecto, lo que se define como ámbito en ingeniería del software.

Este proyecto ha sido propuesto por nuestros tutores, Helena Castán Lanaspa y Salvador Dueñas Carazo, y está orientado al desarrollo de un amplificador Lock-in para extracción de señales a partir de otra distorsionada por el ruido.

Para entender la utilidad de un amplificador Lock-in, imaginemos un dispositivo que, alimentado con una señal sinusoidal, provoca en ésta un desfase. Generalmente, y debido a diversos efectos que se verán posteriormente, las señales están distorsionadas por varios tipos de ruido. Esto representa un problema muy grande si la señal que deseamos medir tiene una amplitud muy pequeña y/o la frecuencia es muy baja. Estas situaciones implican que las componentes de la señal de ruido tendrán amplitudes de incluso varios órdenes de magnitud con respecto a la señal que se desea medir. Esto significa que la relación señal ruido (SNR, *Signal to Noise Ratio*) es muy baja, lo que a su vez provoca que, si se desea utilizar un amplificador, el tanto la señal como el ruido quedarán multiplicados por la ganancia del amplificador, empeorando la situación.

Sin embargo, dado que conocemos la frecuencia de la señal de entrada y su amplitud, es posible realizar una serie de operaciones entre la señal que sale desfasada del circuito y la de entrada a éste, que llamaremos referencia, que permiten rechazar, de la salida, las señales diferentes (en la práctica, a una cierta distancia) de la señal de referencia. También puede emplearse para calcular el desfase producido por el dispositivo bajo prueba, y para calcular la capacitancia y la conductancia del mismo.

En sí, este proyecto es una de las dos partes que componen el proyecto original, que consiste en la implementación software de un sistema que permitiera tres puntos. En primer lugar, que el software funcionara de forma autónoma, como una simulación. En segundo lugar, que utilizando un osciloscopio y un generador de formas de onda, el software fuera capaz de controlar estos para generar la señal, enviarla al generador de formas de onda para que éste las transformara en valores de voltaje, pasarlas por un dispositivo bajo pruebas (DUT, *Device Under Test*), digitalizar los valores de voltaje

o intensidad a la salida a través del generador de señales, y que el sistema pueda separar las componentes de fase y cuadratura, actuando a modo de Lock-in. Una tercera y última parte, que no ha podido ser terminada por restricciones de tiempo, buscaba que el sistema pudiera controlar también el funcionamiento de un Lock-in hardware mediante el puerto GPIB, a través de un conversor USB a GPIB. En esta memoria se presenta el análisis, diseño, e implementación del simulador del sistema formado por el amplificador Lock-in, el generador de señales y el osciloscopio. La otra parte del proyecto, que presentará mi compañero Andrés Escobar Cotán, detalla la conexión y el control de los dispositivos físicos y el generador de ondas.

Sin embargo, estudiando la bibliografía disponible y como se ha comentado en el resumen, casi todos los sistemas Lock-in son dispositivos hardware con un procesador digital de señales; de aquellos que han sido implementados por software que aparecen en la literatura, muchos datan de los años 70-80 o están implementados en LabView. y además, la inmensa mayoría únicamente pueden trabajar en una sola frecuencia. Resulta interesante, desde el punto de vista de la caracterización de circuitos y componentes electrónicos, poder trabajar con barridos o incluso formas de onda complejas, en las que la señal de referencia está formada por varias componentes sinusoidales a la vez. Este aspecto, la caracterización de dispositivos electrónicos, es, como su nombre indica, el objetivo que persigue el grupo de investigación reconocido "Grupo de Caracterización de Materiales y Dispositivos Electrónicos", del que forman parte nuestros tutores.

1.2. Alcance del proyecto

El PMBOK®[1], en su quinta edición, define alcance del producto como "aquellas características y funciones que describen un producto, servicio o resultado", y alcance del proyecto como "el trabajo realizado para entregar un producto, servicio o resultado con las funciones y características especificadas".

El alcance de esta aplicación está centrado en varios grupos. Por una parte, dado que está codificado en el lenguaje C# y admite compilación para Windows, Linux y OS X, puede emplearse directamente como una aplicación de escritorio, teniendo como objetivo grupos de investigación y empresas centradas en el ámbito de la electrónica.

Por otra parte, como se emplea un lenguaje orientado a objetos derivado de C, resulta fácil traducir la aplicación a C++. De esta forma, resulta fácil ejecutarla en microcontroladores recientes, como el PIC32MZ, que dispone, efectivamente de un compilador de C++, suficiente memoria Flash y RAM como para almacenarla (2MB de Flash interna y 512KB de RAM en los modelos superiores, pudiendo aumentarla a través del puerto paralelo maestro o PMP), lo que a su vez permite integrarla en dispositivos que puedan beneficiarse de un amplificador Lock-in, tales como espectrómetros, balanzas de precisión, estabilización de frecuencia en láseres, caracterización de dispositivos microelectrónicos, sensores, etc. Algunas de las optimizaciones tienen como objetivo aumentar la velocidad del sistema y pueden resultar beneficiosas para este tipo de sistemas.

[1] PMBOK es la abreviatura de *Project Management Body of Knowledge*, un libro que recoge una serie de estándares de gestión de proyectos. Ver [82].

El hecho de que esté diseñado de forma modular permite, o bien compilarla a modo de biblioteca y enlazarla con proyectos mayores, o bien incorporando directamente el código desarrollado en este trabajo, tanto en aplicaciones de escritorio como embebidas. De esta forma, puede resultar de utilidad también para investigadores y empresas dedicadas a instrumentación y desarrollo en química analítica, sensores, y en todas aquellas situaciones donde se necesite recuperar una señal en general.

1.3. Objetivos

Los objetivos de este proyecto son varios. Por una parte, los objetivos del proyecto son crear un sistema que pueda simular el comportamiento de un amplificador Lock-in utilizando como lenguaje Matlab, C, o alguno de sus derivados, de tal forma que el diseño esté modularizado. Este sistema debe contar con un generador de funciones, ya sea también virtual o físico, y un osciloscopio para recoger la señal, e igual que el generador, también puede ser virtual o físico. En caso de ser físicos, el sistema debe implementar los drivers para controlarlos, y debe soportar, como mínimo, el osciloscopio Keysight DSO-X 3104A, tanto en su función de generador de señales como de osciloscopio.

El sistema también deberá controlar, a través de puerto GPIB, el amplificador Lock-in SR844 de Stanford Research Systems. Dado que los ordenadores no suelen disponer, normalmente, de puerto GPIB, se utilizará el conversor USB 2.0 a GPIB 82357B de Keysight.

Como se ha introducido en la sección 1.1, el objetivo último del sistema debe ser obtener la componente de fase y cuadratura de la señal resultado de pasar una onda, sintetizada por el generador de funciones, por un dispositivo bajo pruebas. A su vez, de las componente en fase y cuadratura se deben obtener la conductancia y la capacitancia, respectivamente, del dispositivo bajo test.

Las señales generadas por el generador de funciones deben poder ser multifrecuenciales, y se distinguirán tres tipos de señal: ondas sinusoidales simples, que están definidas por una amplitud, una frecuencia y un desfase, y que se repite en el tiempo; ondas compuestas, que es una suma de ondas sinusoidales simples; y barridos de ondas, que van variando de una frecuencia inicial a una final realizando un número constante de ciclos por cada frecuencia intermedia.

De ser posible, el sistema deberá trabajar lo más próximo al tiempo real, optimizando los cálculos para incrementar la velocidad.

Respecto a los objetivos personales, es la primera vez que tanto mi compañero como yo realizamos un proyecto tan centrado en procesado digital de señales. Aunque durante el máster hemos cursado asignaturas relacionadas con la electrónica, el procesado de imágenes (que pueden pensarse como un tipo de señal), y las tecnologías de visualización, como puede ser Sistemas Hardware y Software de Captura y Visualización de Imagen, impartida tanto por nuestros tutores Helena y Salvador como por Javier Finat Codes en la parte de procesado, este proyecto presenta un tema complementario al visto en la asignatura, utilizando técnicas aprendidas en ella o muy similares.

De igual manera, al desconocer gran parte de la materia, supone una oportunidad única para conocer o experimentar el mejor modo de gestionar un proyecto en el que se debe programar a la vez que se profundiza, en base a diversos tipos de fuentes como por ejemplo artículos científicos. Esto supondrá, como se verá más tarde, refactorizar varias veces el código para adaptarlo mejor a las necesidades, realizando varias iteraciones hasta lograr que tanto el modelo de dominio como la interfaz

gráfica sean lo más parecidas posibles a los conceptos y objetos del mundo real.

1.4. Organización de la memoria

Los contenidos de la presente memoria están estructurados de acuerdo al siguiente esquema:

- **Capítulos**

  - El Capítulo 1, **«Introducción»**, expone de forma general cómo surgió el proyecto, sus características generales, en qué grupo de usuarios se ha centrado su desarrollo, los objetivos, y qué diferencia este proyecto con respecto al estado del arte.

  - En el Capítulo 2, **«Fundamentos teóricos»**, se explican los conceptos generales sobre procesado digital de señal, y filtrado y se desarrollan las características y la terminología relacionada a los amplificadores Lock-in en base al modelo SR850 de la compañía Stanford Research Systems. También se expone, en base a diferentes artículos científicos y libros, las aplicaiones de estos sistemas en diversas disciplinas. Finalmente, se presenta el desarrollo matemático en el que se fundamentan este tipo de amplificadores.

  - En el Capítulo 3, **«Gestión del proyecto»**, se muestran la planificación estimada y la que finalmente se ha seguido de cara al desarrollo del mismo, la metodología de trabajo, las personas que han participado en él, y los costes que habría tenido, en material software y hardware y horas de trabajo. También se evalúan los riesgos que a priori presentaba la ejecución y se describen las diferentes herramientas de trabajo empleadas.

  - En el Capítulo 4, **«Análisis»**, se enumeran los requisitos del proyecto, sus casos de uso principales y los diagramas de secuencia más importantes.

  - El Capítulo 5, **«Diseño»**, se explica cómo se interrelacionan y agrupan en paquetes los diferentes componentes y clases obtenidos en la fase anterior y cómo se comunican entre sí los diferentes plataformas y elementos hardware y los entornos de ejecución. También se detallan los casos de uso detectados en la fase de análisis, y se muestra el proceso de desarrollo de la interfaz gráfica.

  - El Capítulo 6, **«Implementación, pruebas y resultados»**, se detalla el procedimiento seguido para transformar los diagramas resultantes de las fases previas en código ejecutable. Se exponen también los algoritmos que se emplean, sus ventajas e inconvenientes, los problemas que se han encontrado y sus soluciones. A la vez, se comenta la batería de pruebas que se ha realizado, tanto de la aplicación en general como de sus componentes, y las pruebas con dispositivos físicos. También se discuten las ptuebas de adecuación de la interfaz con diferentes personas del grupo objetivo del desarrollo.

  - Por último el Capítulo 7, **«Conclusiones»**, se comparan los resultados obtenidos con los objetivos, y se resume de forma global lo que este proyecto ha supuesto tanto en términos de conocimiento como de preparación para casos similares. También se incluyen algunas

de las maneras en las que el este trabajo puede extenderse, algunos de los errores que deben corregirse y aspectos pendientes de investigar e implementar.

- **Apéndices**

  - **Apéndice A, Glosario.** Enlista la terminología necesaria para una correcta comprensión del proyecto, así como sus definiciones.

  - **Apéndice B, Contenido del CD-ROM.** En este apéndice se detallan los directorios y los ficheros que pueden encontrarse en el disco adjunto a esta memoria; entre ellos se encuentran las referencias relacionadas con en el proyecto, el código fuente y algunas de las pruebas.

Capítulo 2
Fundamentos teóricos

"Cree que puedes y estarás a mitad de camino"

Theodore Roosevelt

Dado que este proyecto trabaja con conceptos propios de teoría de la señal y procesado digital de señales, conviene introducir, antes de explicar cómo se ha desarrollado el proyecto, los conceptos básicos relacionados con él.

En este capítulo veremos, además, los fudamentos de los amplificadores Lock-in, las aplicaciones que a lo largo de los últimos años ha encontrado, y el desarrollo matemático de las ecuaciones que explican y rigen estos amplificadores.

2.1. Introducción a las señales

Hemos comentado en la introducción, que este proyecto está relacionado con las señales, pero ¿qué es una señal?

Una **señal** es simplemente, la variación de una magnitud física con respecto de otra. Así, tenemos señales eléctricas, lumínicas, sonoras,... Es precisamente esta variación la que hace que las señales puedan transmitir información. Por ejemplo, la variación en la intensidad lumínica puede emplearse para transmitir un mensaje por código Morse, o la variación eléctrica entre los extremos de un termopar puede informarnos acerca de la temperatura.

El objetivo de los **sensores** es capturar esas señales. Son dispositivos capaces de convertir esas variaciones de una o varias magnitudes físicas (o químicas) en variaciones de voltaje o intensidad, por ejemplo.

Una **onda** es un fenómeno vibratorio consistente en la propagación de una perturbación de alguna propiedad o magnitud física, a través de un medio. Existen ondas **periódicas**, como por ejemplo las ondas lumínicas y sonoras y *no periódicas*, como puede ser un pulso, una onda exponencial, o algún tipo de ruido aleatorio. A su vez, dentro de las ondas periódicas, existen varias clases; como son las sinusoidales, dientes de sierra, triangulares, etc.

Las ondas sinusoidales son las ondas más comunes, y un ejemplo de éstas se puede ver en la figura 2.1. En ella aparecen marcadas las características de las ondas, como son la **amplitud**, que se define como la diferencia ente el valor más alto que alcanza (esto es, el pico), y su punto medio; la

amplitud pico a pico, que es la distancia que existe entre los valores de un pico y un valle, y es el doble de la amplitud; el **periodo**, que se refiere al tiempo en el que la onda completa una oscilación, y su inverso es la **frecuencia**, definida como el número de oscilaciones que la onda realiza por segundo.

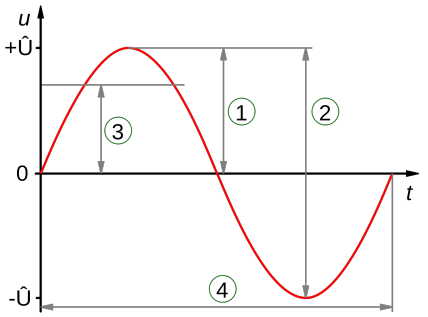
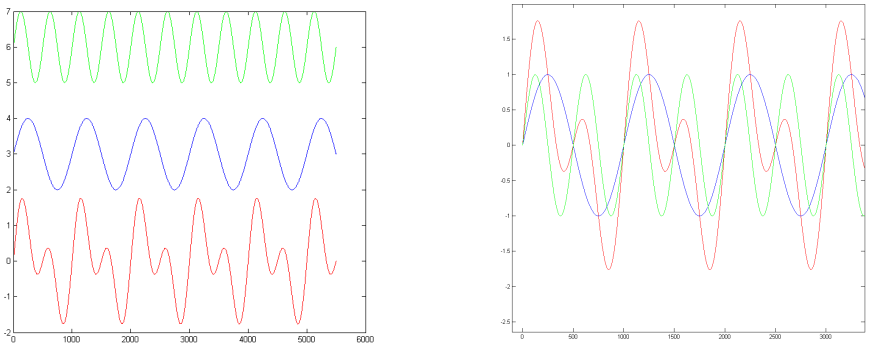


Figura 2.1: Un ciclo de una onda senoidal, con algunas de sus características etiquetadas. 1.- Amplitud, 2.- Amplitud pico a pico, 3.- Media cuadrática (RMS), 4.- Periodo. Tomado de [6].

Las **formas de onda compuestas** se crean cuando se transmite más de una señal a la vez por un canal, con lo cual se suman. Por ejemplo, si, como podemos ver en la figura 2.2 generamos dos ondas sinusoidales, una de 1 Hz (representada en azul), y otra de 2 Hz (representada en verde), y las sumamos sus valores entre sí punto a punto, obtendremos la onda representada en rojo. Es importante destacar que la frecuencia de la onda compuesta es de 1Hz (es decir, sus valores se repiten a la vez que los de la original azul). Decimos entonces que la onda compuesta tiene una frecuencia **fundamental** de 1 Hz, y un **armónico** al doble de la frecuencia fundamental.



(a) Ondas separadas

(b) Ondas superpuestas

Figura 2.2: Composición de ondas. La onda representada en azul vibra a 1Hz, la verde lo hace a 2Hz, y la suma de las dos está representada en color rojo.

2.1.1. Amplificador Lock-in

Una forma primitiva de amplificador Lock-in fue inventada a principios de la década de los treinta por C. R. Cosens [54], y posteriormente fabricada por Cambridge Instrument Corporation [51]. Inicialmente se trataba de un detector de equilibro para puentes de AC, que era sensible únicamente a las frecuencias aplicadas a las impedancias que debían ser balanceadas, y por tanto aún no recibía la denominación de Lock-in. Durante esa década, el concepto se empezó a aplicar en los campos de la astrofísica y de la geofísica para eliminar las componentes de ruido de las señales, por ejemplo para observar la corona solar durante el día. Esta idea fue desarrollada por Walter C. Michels y Norma L. Curtis a principios de la década de los cuarenta, que en su diseño aplicaban la señal de referencia a las rejillas pantalla de dos pentodos. Posteriormente, Robert Henry Dicke, mientras trabajaba para el Radiation Laboratory del M.I.T., continuó desarrollando este instrumento basándose en el trabajo de Michels y Curtis, y terminaría fundando la compañía Princeton Applied Research [23], dedicada a comercializar amplificadores Lock-in.

La funcionalidad principal que tienen los Lock-in es que, como hemos comentado anteriormente, son capaces de eliminar todas las componentes no deseadas de una señal cuando se dispone de una señal limpia a la misma frecuencia que la componente de la señal ruidosa que se desea conservar, funcionando como un filtro pasabanda con un ancho de banda muy pequeño. Veremos más sobre este tema en la sección 2.2.

Figura 2.3: Amplificador Lock-in SR844 y generadores de señales TG210 y 33120A, en el laboratorio 1L012.

2.1.2. Osciloscopio

Los *osciloscopios* son instrumentos que permiten ver las variaciones de una señal eléctrica, generalmente voltaje, en el tiempo, mediante su representación en una pantalla. De esta manera, el tiempo es el eje horizontal y el voltaje el vertical.

Inicialmente eran analógicos, y se basaban en un tubo de rayos catódicos (CRT). En ellos existe un cañón de electrones en el fondo del tubo; una fuente de voltaje de corriente continua se conecta entre el ánodo y el cátodo. El cátodo se calienta mediante una pequeña bobina y se desprenden electrones por termoemisión, que son acelerados hacia el ánodo. La señal que desea ser visualizada V pasa por un amplificador y controla las bobinas que provocan la deflexión vertical del haz de electrones proveniente del cañón. A las bobinas de deflexión horizontal se les aplica una señal tipo diente de sierra. De esta forma, en ausencia de la señal a visualizar V la pantalla CRT produce una línea horizontal en el centro de la pantalla provocada por el diente de sierran. Cuando se conecta la señal V al canal de entrada del osciloscopio, el haz se desvía, con respecto al centro, proporcionalmente al valor de voltaje de V.

Los osciloscopios modernos son digitales, y utilizan procesado digital de señales. Esto significa que un circuito llamado **conversor analógico digital**, conversor A/D, o DAC (siglas de su nombre en inglés *Analog-to-Digital Converter* transforma el valor de voltaje de canal en un número binario, que posteriormente procesa un microcontrolador o microprocesador.

Muchos osciloscopios tienen más de un canal, lo que les permite mostrar varias señales en la pantalla a la vez. Por ejemplo, el osciloscopio con el que está desrrollado este trabajo fin de máster, el DSO-X 3104A de Keysight, dispone de cuatro canales, cada uno representado en la pantalla por un color diferente.

En este tipo de instrumentos existe un sistema llamado *trigger*, que permite 'congelar' la imagen de la pantalla cuando uno de los canales supera un valor de voltaje, que puede ajustarse. De esta forma, es posible ver señales no periódicas como por ejemplo pulsos en un bus.

2.1.3. Generador de ondas

Estos dispositivos permiten generar señales con la posibilidad de definir sus parámetros, como son amplitud, frecuencia, y componente de continua. También se puede elegir el tipo de forma de onda, que suele ser senoidal, cuadrada, triangular o diente de sierra. Las versiones de alta gama permiten generar señales multifrecuenciales y mezclarlas. Veremos más adelante un ejemplo en la figura 5.6.

2.2. Fundamentos del amplificador Lock-in

En esta sección expondremos el funcionamiento interno y la terminología y conceptos relacionados con los amplificadores Lock-in, que serán necesarios a la hora de realizar una implementación del mismo como una aplicación software.

Los siguientes apartados están basados en el el texto de [88].

2.2.1. ¿Qué es un amplificador Lock-in?

Los amplificadores Lock-In se utilizan para detectar y medir señales AC muy pequeñas -de hasta unos pocos nanovoltios-. Estos aparatos permiten realizar mediciones precisas incluso cuando la pequeña señal está distorsionada por fuentes de ruido varios cientos de veces mayores.

Los amplificadores lock-in utilizan una técnica conocida como detección sensible a la fase para separar el componente de la señal a una frecuencia y fase de referencia específicas. Las señales de ruido a frecuencias diferentes a la de referencia son rechazadas y no afectan a la medición.

¿Por qué usar un Lock-in?

Consideremos por ejemplo que se desea amplificar una señal onda sinusoidal de 10 nV a 10kHz. Un buen amplificador de bajo ruido tendrá en torno a 5 $\frac{\text{nV}}{\sqrt{\text{Hz}}}$ de ruido de entrada. Si el ancho de banda del amplificador es de 100 kHz y la ganancia es 1000, entonces podemos esperar que la salida sea de 10 μV de señal (10 nV x 1000) y 1.6 mV de ruido de banda ancha (5 $\frac{\text{nV}}{\sqrt{\text{Hz}}} \times \sqrt{100}$ kHz $\times$ 1000). Será difícil medir la señal de salida sin separar la señal de interés.

Si añadimos a la salida del amplificador un filtro pasa banda con una Q = 100 (un muy buen filtro) centrado en 10kHz, cualquier señal en un ancho de banda de 100 Hz será detectada (10 kHz/Q). El ruido en el filtro pasa banda será de 50 μV (5 $\frac{\text{nV}}{\sqrt{\text{Hz}}} \times \sqrt{100}$ Hz $\times$ 1000) y la señal seguirá siendo de 10μV. El ruido de salida es mucho mayor que la señal y no se puede realizar una medición precisa. Una mayor ganancia no ayudará al problema de la relación señal-ruido.

Ahora intentemos añadir a la salida del amplificador un detector sensible a la fase (PSD). ¡El PSD puede detectar la señal a 10 kHz con un ancho de banda tan estrecho como 0.01 Hz! En este caso, el ruido en el ancho de banda de detección será de tan sólo 0.5 μV (5 nV/ 0.01 Hz x 1000) mientras que la señal sigue siendo de 10 μV. La relación señal-ruido es ahora de 20 y es posible una medición precisa de la señal.

¿Qué es la detección sensible a la fase?

Las mediciones en un Lock-In necesitan una frecuencia de referencia. Típicamente un experimento se excita a una frecuencia dada (utilizando un oscilador o generador de funciones) y el lock-in detecta la respuesta del experimento a la frecuencia de referencia. En el diagrama de la Figura 2.4, la señal de referencia es una onda cuadrada a la frecuencia r. Esta podría ser la salida de sincronización de un generador de funciones. Si la salida sinusoidal del generador de funciones se utiliza para excitar el experimento, la respuesta podría ser la forma de onda que se indica en la parte inferior de esa misma figura. La señal es $V_{SIG} \cdot sin(\omega_R t + \theta_{SIG})$ donde V_{SIG} es la amplitud de la señal.

El modelo SR850 genera su propia onda sinusoidal, mostrada debajo como referencia del lock-in. La referencia del lock-in es $V_L \cdot sin(\omega_L t + \theta_{REF})$.

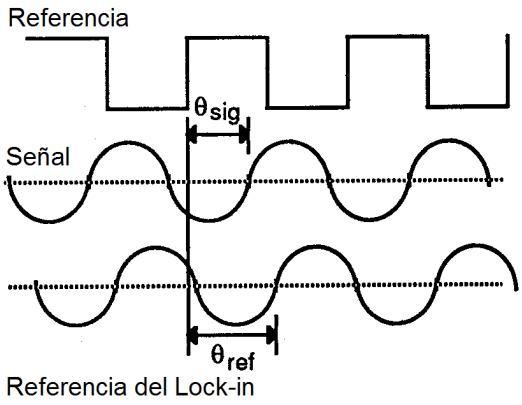


Figura 2.4: Relación entre la referencia del Lock-in, la señal y la referencia.

El SR850 amplifica la señal y la multiplica entonces por la referencia del lock-in utilizando un detector sensible a la fase o multiplicador. La salida del PSD es simplemente el producto de dos ondas

sinusoidales.

$$V_{PSD} = V_{SIG}V_L sin(\omega_r t + \theta_{SIG}) sin(\omega_L t + \theta_{REF}) \tag{2.1}$$

$$= \frac{1}{2}V_{SIG}V_L cos([\omega_R - \omega_L]t + \theta_{SIG} - \theta_{REF}) - \frac{1}{2}V_{SIG}V_L cos([\omega_R + \omega_L]t + \theta_{SIG} + \theta_{REF}) \tag{2.2}$$

La salida del PSD consiste en dos señales en AC, una a la diferencia de frecuencias ($\omega_R - \omega_L$) y otra a la suma ($\omega_R + \omega_L$).

Si la salida del PSD se pasa a través de un filtro pasa baja, las señales en AC quedan eliminadas. ¿Qué quedaría? Generalmente, nada. Sin embargo, si ω_R es igual a ω_L, la componente de diferencia de frecuencias será una señal en DC. En este caso, la salida filtrada por el PSD será

$$V_{PSD} = \frac{1}{2}V_{SIG}V_L cos(\theta_{SIG} - \theta_{REF}) \tag{2.3}$$

Esta señal es una señal continua proporcional a la amplitud de la señal.

Detección en banda estrecha

Supongamos ahora que la entrada está afectada por ruido. El PSD y el filtro pasa baja únicamente detectan señales cuyas frecuencias son muy cercanas a la frecuencia de referencia del lock-in. Las señales de ruido a frecuencias lejanas a la de referencia quedan atenuadas a la salida del PSD por el filtro pasa baja (ni $\omega_{NOISE} - \omega_{REF}$ ni $\omega_{NOISE} + \omega_{REF}$ son cercanas a DC). El ruido a frecuencias muy cercanas a la de referencia quedará como señales AC de baja frecuencia a la salida del PSD ($|\omega_{NOISE} - \omega_{REF}|$ es pequeño). Su atenuación depende del ancho de banda y del *roll-off*[1] del filtro pasa baja. Un ancho de banda más reducido eliminará las fuentes de ruido muy cercanas a la frecuencia de referencia, un ancho de banda amplio permitirá el paso de esas señales. El ancho de banda del filtro pasa baja determina el ancho de banda de la detección. Sólo la señal a la frecuencia de referencia dará lugar a una auténtica salida en DC y quedará inalterada por el filtro pasa baja. Esta es la señal que queremos medir.

¿De dónde proviene la referencia del Lock-in?

Necesitamos hacer que la señal de referencia del lock-in tenga la misma frecuencia que la de la señal, esto es, $\omega_R = \omega_L$. No solo han de ser iguales las frecuencias; la fase entre las señales no puede cambiar con el tiempo, de otra forma $cos(\theta_{SIG} - \theta_{REF})$ variará y V_{PSD} no será una señal en DC. En otras palabras, la señal del lock-in necesita estar enganchada (sincronizada) en fase a la señal de referencia.

Los amplificadores lock-in utilizan un bucle de enganche de fase (PLL, *phase-locked loop*) para generar la señal de referencia. Se aporta al lock-in una señal de referencia externa (en este caso, la onda cuadrada de referencia). El PLL del lock-in engancha el oscilador de referencia interno a esta referencia externa, dando lugar a una onda sinusoidal de referencia oscilando a ω_R con un desfase fijo de θ_{REF}. Ya que el PLL vigila activamente la referencia externa, los cambios en la frecuencia de esta última no afectan a la medición.

[1] El `roll-off` de un filtro se refiere a la pendiente que existe entre la banda de paso y la banda de rechazo, o lo que es lo mismo, la pendiente de la banda de transición. Ver [90], sección *"Frequency Response of a 1st-order Low Pass Filter"* y [69], página 37.

Todas las mediciones del lock-in necesitan una señal de referencia

La referencia puede provenir de una fuente de excitación (por ejemplo, un generador de funciones), en cuyo caso se trata de una señal de referencia externa. También es posible utilizar el oscilador interno del SR850 en su lugar. El oscilador interno es igual que un generador de funciones (con salida sinusoidal variable y señal de sincronización TTL) que está sincronizado en fase con el oscilador de referencia.

Fase y magnitud

La salida del PSD es proporcional a $V_{SIG}cos(\theta)$ donde $\theta = (\theta_{SIG} - \theta_{REF})$. θ es la diferencia entre la fase de la señal y la del oscilador de referencia del lock-in. Podemos hacer θ igual a cero ajustando θ_{REF}, en cuyo caso podremos medir V_{SIG} ($cos(\theta) = 1$). Por el contrario, si θ es de 90°, no habrá salida alguna. Se dice que un lock-in es monofásico si cuenta con un único PSD, y su salida es $V_{SIG}cos(\theta)$.

Esta dependencia de la fase puede ser eliminada acoplando un segundo PSD. Si el segundo PSD multiplica la señal con la del oscilador de referencia desfasada 90°, esto es, $V_L sin(\omega_L t + \theta_{REF} + 90)$, su salida con el filtrado pasa baja será

$$V_{PSD2} = \frac{1}{2} V_{SIG} V_L sin(\theta_{SIG} - \theta_{REF}) \tag{2.4}$$

$$V_{PSD2} \sim V_{SIG} sin(\theta) \tag{2.5}$$

Ahora disponemos de dos salidas, una proporcional a $cos(\theta)$ y otra proporcional a $sin(\theta)$. Si llamamos X a la primera salida e Y a la segunda, estas dos cantidades representan la señal como un vector relativo al oscilador de referencia del lock-in. X es la componente en fase e Y la componente en cuadratura. Esto es debido a que cuando $\theta = 0$, X mide la señal mientras que Y es cero.

La dependencia de la fase se elimina calculando la magnitud (R) de la señal vectorial.

$$R = \sqrt{X^2 + Y^2} = V_{SIG} \tag{2.6}$$

R mide la amplitud de la señal y no depende del desfase entre la señal y la referencia del lock-in.

Un lock-in bifásico, como el SR850, tiene dos PSDs, con osciladores de referencia desfasados 90º uno respecto al otro, y puede medir X, Y, y R directamente. Además, se puede medir la fase θ entre la señal y la referencia del lock-in, de acuerdo con:

$$\theta = tan^{-1}\left(\frac{Y}{X}\right) \tag{2.7}$$

2.2.2. ¿Qué mide un Lock-in?

Según el teorema de Fourier, cualquier señal de entrada puede ser representada como la suma de muchas ondas sinusoidales de diferentes amplitudes, frecuencias y fases. A esto se le denomina representar una señal en el dominio de la frecuencia. Los osciloscopios corrientes muestran la señal en el 'dominio del tiempo'. Exceptuando el caso de las ondas sinusoidales simples, la representación en el dominio del tiempo no aporta mucha información sobre las diferentes frecuencias que conforman la señal.

¿Qué mide el SR850?

El SR850 multiplica la señal por una onda sinusoidal pura a la frecuencia de referencia. Todas las componentes de la señal de entrada son multiplicadas por la referencia simultáneamente. Matemáticamente hablando, las ondas sinusoidales de diferentes frecuencias son ortogonales, esto es, la media del producto de dos ondas sinusoidales es cero a menos que sus frecuencias sean exactamente la misma. En el SR850, el producto de esta multiplicación da una señal de salida en DC proporcional a la componente de la señal cuya frecuencia está estrictamente enganchada a la frecuencia de referencia. El filtro pasa baja que hay a continuación del multiplicador produce el promediado que elimina los productos de la referencia con las componentes de la señal de entrada a todas las demás frecuencias.

El SR850, puesto que multiplica la señal con una onda sinusoidal pura, mide la única componente de Fourier (seno) de la señal a la frecuencia de referencia. Como ejemplo consideremos que la señal de entrada es una onda cuadrada simple a la frecuencia f. La onda cuadrada está, de hecho, compuesta por múltiples ondas sinusoidales a múltiplos de f con fases y amplitudes cuidadosamente relacionadas. Se puede expresar una onda cuadrada de 2V pico a pico como

$$S(t) = 1{,}273sin(\omega t) + 0{,}4244sin(3\omega t) + 0{,}2546sin(5\omega t) + \ldots \tag{2.8}$$

donde $\omega = 2\pi f$. El SR850, enganchado a f, separará la primera componente. La señal medida será $1{,}273sin(\omega t)$, no los 2V pico a pico que proporcionaría un osciloscopio.

En el caso más general, la entrada está formada por la señal y ruido. Este último se representa como señales a todas las frecuencias. El lock-in ideal es únicamente sensible al ruido a la frecuencia de referencia. El ruido a otras frecuencias queda eliminado por el filtro pasa baja ubicado a continuación del multiplicador. Esta reducción del ancho de banda es la ventaja principal que ofrece un amplificador lock-in. Únicamente las entradas a frecuencias iguales a la frecuencia de referencia darán lugar a una salida.

¿RMS o pico a pico?

Como regla general, los amplificadores lock-in muestran la señal de entrada en voltios RMS[2]. Cuando el SR850 muestra una magnitud de 1V (rms), la componente de la señal de entrada a la frecuencia de referencia es una onda sinusoidal con una amplitud de 1V (rms) o 2.8 V pico a pico.

Por tanto, en el ejemplo anterior con una señal cuadrada de entrada de 2V pico a pico, el SR850 detectaría la primera componente sinusoidal, $1{,}273sin(\omega t)$. La magnitud medida y mostrada sería 0.90 V (rms), $(\frac{1}{\sqrt{2}} \cdot 1{,}273)$.

¿Grados o radianes?

Hasta ahora las frecuencias han sido referenciadas como f (Hz) y como ω ($2\pi f$ radianes/segundo). Esto se debe a que las frecuencias se miden en ciclos por segundo y las matemáticas trabajan mejor en radianes. Para propósitos de medida, las frecuencias tal y como se miden en un amplificador lock-in están en el rango de los Hz. Las ecuaciones utilizadas para explicar los propios cálculos están a veces escritas utilizando ω para simplificar las expresiones.

La fase siempre se representa en grados. De nuevo, esto es más por costumbre que por elección. Las ecuaciones escritas como $sin(\omega t + \theta)$ están representadas como si θ estuviera en radianes, principalmente por sencillez. Los amplificadores lock-in siempre miden y operan con la fase en grados.

[2] RMS es el acrónimo de *Root Mean Square*, valor cuadrático medio.

2.2.3. El SR850 funcional

En la Figura 2.5 se muestra el diagrama de bloques funcionales del Amplificador Lock-in DSP SR850. Las funciones dentro del área gris son llevadas a cabo por el procesador digital de señal (DSP, digital signal processor). Comentaremos los aspectos del DSP del SR850 a medida que vayan surgiendo en la descripción de cada bloque funcional.

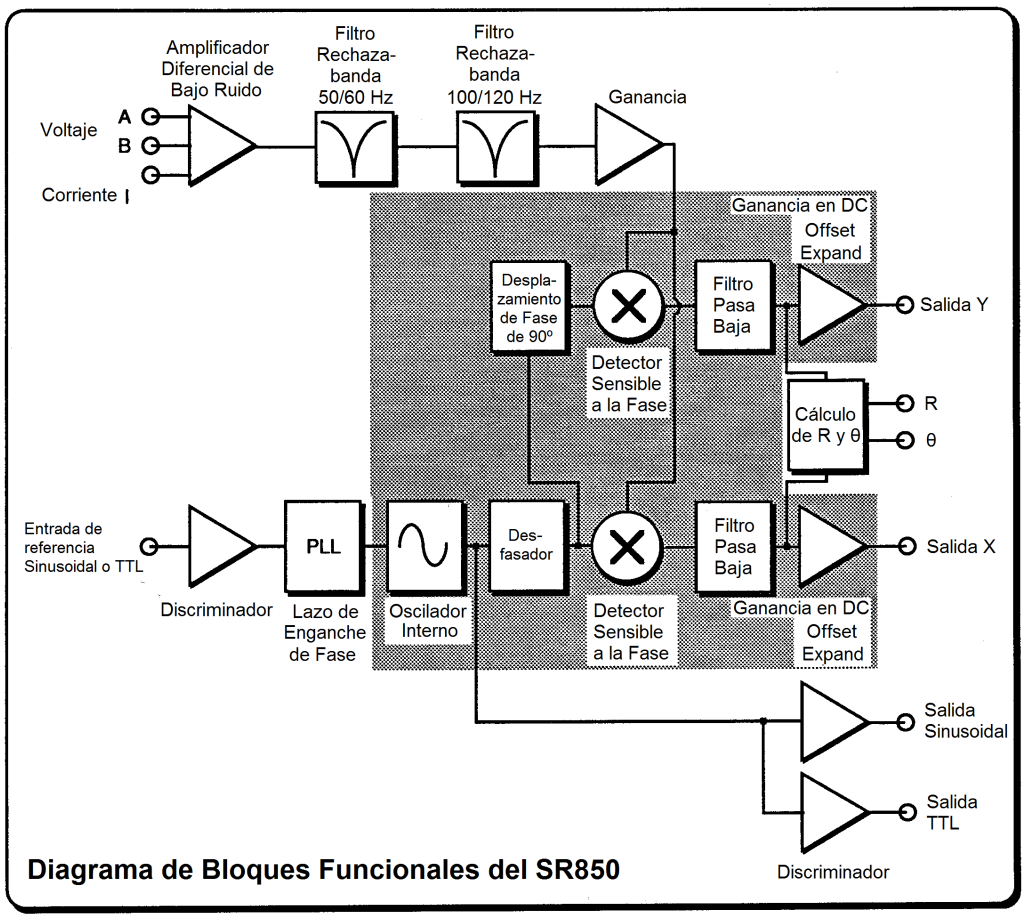


Figura 2.5: Diagrama de bloques funcionales del SR850

2.2.4. El canal de referencia

Un amplificador lock-in necesita un oscilador de referencia enganchado en fase a la frecuencia de la señal. En general, esto se logra sincronizando en fase un oscilador interno a una señal de referencia aportada exteriormente. Esta señal de referencia a menudo proviene de la fuente de señal que aporta la excitación al experimento.

Entrada de referencia

La entrada de referencia del SR850 puede aceptar una señal analógica (como una onda sinusoidal), o una señal lógica TTL. Al primer caso se lo denomina 'Seno externo'. La entrada está acoplada en AC (por encima de 1 Hz) y la impedancia de entrada es de 1 MΩ. Una señal sinusoidal de entrada mayor que 200 mV pico a pico activará el discriminador de la entrada. Se detectan los pasos por cero con pendiente positiva, que son considerados como cero para el desfase de referencia.

Se pueden utilizar señales de referencia TTL a cualquier frecuencia hasta 102 kHz. Para frecuencias inferiores a 1 Hz, es obligatoria una señal de referencia TTL. Muchos generadores de funciones proporcionan una salida TTL de sincronización (SYNC) que puede ser utilizada como referencia. Esto resulta útil, ya que la salida sinusoidal del generador podría ser inferior a 200 mV o variar en amplitud. La señal de sincronización aportará una referencia estable con independencia de la amplitud del seno. Cuando se emplee una referencia TTL, la referencia de entrada puede ajustarse como 'External Rising' (detección de flancos de subida o ascendentes) o 'External Falling' (detección de flancos de bajada o descendentes). En cada caso, el oscilador interno está enganchado con desfase cero al flanco detectado.

El oscilador interno

El oscilador interno del SR850 es básicamente un generador de funciones de 100 kHz con salidas sinusoidal y de sincronización TTL. El oscilador puede engancharse a la fase de la referencia externa.

El oscilador genera una onda sinusoidal sintetizada digitalmente. El procesador digital de señales, o DSP (digital signal processor) envía los valores calculados del seno a un conversor digital analógico cada cuatro μs (256 kHz). Un filtro de suavizado convierte la señal muestreada en una onda sinusoidal poco distorsionada. La onda sinusoidal del oscilador interno sale por el conector BNC 'SINE OUT' en el panel frontal. Se puede ajustar la amplitud de esta salida entre 4mV y 5V.

Cuando se emplea una referencia interna, esta onda sinusoidal del oscilador interno se engancha a la fase de la referencia. El paso por cero ascendente está enganchado al paso por cero o flanco detectado. En este modo, la salida SINE OUT proporciona una onda sinusoidal enganchada en fase a la referencia externa. A bajas frecuencias (inferiores a 10 Hz) el enganche en fase es llevado a cabo digitalmente por el DSP. A frecuencias mayores, se emplea un comparador de fase discreto.

El oscilador interno puede emplearse sin una referencia externa. En el modo 'Internal Reference' (referencia interna), la salida SINE OUT proporciona la excitación para el experimento. El bucle de enganche de fase no se utiliza en este modo ya que la referencia del lock-in está aportando la señal de excitación.

La salida 'TTL OUT' en el panel frontal suministra una salida de sincronización TTL. Los pasos por cero ascendentes del oscilador interno son detectados y convertidos a niveles TTL. La salida es una onda cuadrada.

Osciladores de referencia y fase de referencia

La onda sinusoidal del oscilador interno no es la señal de referencia para los detectores sensibles a la fase. El DSP calcula una segunda onda sinusoidal, desplazada θ_{REF}, a partir del oscilador interno (y por tanto, a partir de una referencia externa), como la referencia de entrada al detector sensible a la fase de X. Esta onda es $sin(\omega_R t + \omega_{REF})$. El desplazamiento de fase de referencia puede ajustarse en incrementos de 0.001º. La entrada al PSD de Y es una tercera onda sinusoidal, calculada por el DSP, desplazada 90º a partir de la segunda onda sinusoidal. Esta onda es $sin(\omega_R t + \theta_{REF} + 90^\circ)$. Ambas ondas sinusoidales de referencia son calculadas con una precisión de 20 bits y se calcula un nuevo

punto cada 4 μs (256 kHz). Los desplazamientos de fase (θ_{REF} y el desfase de 90º) son también números exactos y con una precisión superior a 0.001º. Ninguna de las formas de onda anteriores sale en forma analógica ya que los detectores sensibles a la fase son, de hecho, instrucciones de multiplicación dentro del DSP.

Fluctuaciones de fase

Cuando se emplea una referencia externa, el bucle de enganche de fase añade pequeñas fluctuaciones de fase (*phase jitter*). Se supone que el oscilador interno está enganchado con desfase cero a la referencia externa. El concepto de fluctuaciones de fase se refiere a que el desfase medio es cero pero el desfase instantáneo tiene unos pocos miligrados de ruido. Esto aparece en la salida como ruido en las medidas de fase o cuadratura.

El ruido de fase también puede provocar la aparición de ruido en las salidas X e Y. Esto se debe a que un oscilador de referencia con mucho ruido de fase es equivalente a una referencia cuyo espectro de frecuencias se ha extendido. Esto es, la referencia no es una única frecuencia, sino una distribución de frecuencias en torno a la auténtica frecuencia de referencia. Estas frecuencias espurias están bastante atenuadas pero aún pueden causar problemas. Las frecuencias de referencia espurias dan lugar la detección de señales cercanas a la de referencia.

El ruido de fase en el SR850 es muy bajo y generalmente no provoca problemas. En aplicaciones que necesiten la ausencia de ruido de fase se debe utilizar el modo de referencia interna. Dado que no hay PLL, el oscilador interno y las ondas sinusoidales de referencia están directamente enlazadas y no hay fluctuaciones en la fase medida. (De hecho, las fluctuaciones de fase son el ruido de fase de un oscilador de cuarzo y son extraordinariamente pequeñas).

Detección de armónicos

Es posible calcular las dos ondas sinusoidales de referencia para los PSDs a un múltiplo de la frecuencia del oscilador interno. En este caso, el lock-in detecta señales a $N \times f_{REF}$ que sean síncronas con la referencia. La frecuencia de la salida SINE OUT no queda afectada. El SR850 puede detectar cualquier armónico hasta $N = 32767$ mientras $N \times f_{REF}$ no exceda 102 kHz.

2.2.5. Los detectores sensibles a la fase (PSDs)

El SR850 multiplica la señal con las ondas sinusoidales de referencia digitalmente. La señal amplificada se convierte a digital utilizando un convertidor analógico digital muestreando a 256 kHz. El conversor analógico digital está precedido de un filtro de suavizado de 102 kHz para evitar el evitar que frecuencias superiores sufran rizado por debajo de 102 kHz. El amplificador de señal y los filtros se discutirán posteriormente. El flujo de datos entrada se multiplica, un valor cada vez, con las ondas sinusoidales de referencia calculadas descritas anteriormente. Cada 4 μs se muestrea la señal de entrada y el resultado se multiplica por las dos ondas sinusoidales de referencia (desfasadas 90º una respecto a la otra).

Los detectores sensibles a la fase (PSDs) del SR850 funcionan como multiplicadores lineales, esto es, multiplican la señal por la onda sinusoidal de referencia. Los PSDs analógicos (tanto de onda cuadrada como lineales) tienen muchos problemas asociados a ellos. Los problemas principales son el rechazo de armónicos, offsets de salida, reserva dinámica limitada y error de ganancia.

Por su parte, el PSD digital multiplica la señal digitalizada con una onda sinusoidal de referencia calculada. Dado que las ondas de referencia sinusoidales se calculan con una precisión de 20 bits, contienen muy pocos armónicos. De hecho, ¡los armónicos están al nivel de -120 dB! Esto significa

que la señal se multiplica por una única onda sinusoidal de referencia (en lugar de por una referencia y sus múltiples armónicos) y únicamente la señal en esta única frecuencia de referencia es detectada. El SR850 es completamente insensible a las señales en armónicos de la referencia. Por el contrario, un lock-in que multiplique una onda cuadrada detectará todos los armónicos impares de la referencia (una onda cuadrada contiene un gran número de armónicos impares.

El offset de salida es un problema porque la señal de interés es una salida en DC del PSD y un offset de salida contribuye al error y a una desviación del cero (*zero drift*). Los problemas de offset de los PSDs analógicos quedan eliminados utilizando el multiplicador digital. No hay offsets de salida en DC erróneos de la multiplicación digital de la señal y la referencia. De hecho, la propia multiplicación está totalmente libre de errores. La reserva dinámica de un PSD analógico está limitada en torno a 60 dB. Cundo está presente una señal de ruido elevada, 1000 veces o 60 dB mayor que el rango completo de la señal, el PSD analógico mide la señal con un error. El error está provocado por la no-linealidad en la multiplicación (el error en la salida depende de la amplitud de la entrada). Este error puede ser bastante grande (10 % del rango completo) y depende de la amplitud, frecuencia y forma de onda del ruido. Dado que en el ruido varían bastante los parámetros anteriores, el error del PSD provoca bastante incertidumbre.

En un lock-in digital, la reserva dinámica queda limitada por la calidad de la conversión A/D. Una vez que la señal de entrada es digitalizada, no se introducen más errores posteriormente. Sin duda, la precisión de la multiplicación no depende del tamaño de los números. El conversor A/D utilizado en el SR850 es extremadamente lineal, lo que significa que la presencia de grandes señales de ruido no afecta a su capacidad de digitalizar correctamente una señal pequeña. De hecho, la reserva dinámica del SR850 puede superar los 100 dB sin ningún problema. Hablaremos más sobre reserva dinámica un poco después.

Un PSD lineal analógico multiplica la señal por una onda sinusoidal analógica de referencia. Cualquier variación en la amplitud de referencia aparece directamente como una variación en la ganancia global. Los generadores de ondas sinusoidales analógicos son susceptibles a la variación en la amplitud, especialmente como función de la temperatura. Las ondas sinusoidales de referencia digitales tienen una amplitud precisa y ésta nunca cambia. Esto elimina una de las principales fuentes de error en la ganancia en un lock-in analógico lineal.

El rendimiento global de un amplificador lock-in queda determinado fundamentalmente por el rendimiento de sus detectores sensibles a la fase. En prácticamente todos los aspectos, los PSDs digitales superan a sus homólogos analógicos.

Hemos discutido cómo el procesador digital de señal en el SR850 calcula el oscilador interno y dos ondas sinusoidales de referencia y controla ambos detectores sensibles a la fase. En la siguiente sección, veremos cómo el mismo DSP lleva a cabo el filtrado pasa baja y la amplificación DC necesaria en la salida de los PSDs. Aquí, de nuevo, la técnica digital elimina muchos de los problemas asociados a los amplificadores lock-in analógicos.

2.2.6. Constantes de tiempo y ganancia en DC

Como se ha dicho, la salida del PSD contiene múltiples señales. La mayoría de las señales de salida tienen frecuencias que son o bien la suma, o bien la diferencia, entre la frecuencia de una señal de entrada y la frecuencia de referencia. Únicamente la componente de la señal de entrada cuya frecuencia sea exactamente igual a la frecuencia de referencia dará lugar a una salida en DC.

El filtro pasa baja a la salida del PSD elimina todas las señales AC indeseadas, tanto la 2F (suma de la señal y la referencia) como las componentes de ruido. Este filtro es lo que hace al lock-in un detector de banda tan estrecha.

Constantes de tiempo

Los amplificadores lock-in tradicionalmente han ajustado el ancho de banda del filtro pasa baja mediante la configuración de la constante de tiempo. La constante de tiempo es simplemente $\frac{1}{2\pi f_C}$, donde f_C es la frecuencia de corte[32], esto es, la frecuencia del filtro a la cual la amplitud de la señal de entrada se atenúa en 3 dB, o dicho de otro modo, el polo del filtro a -3 dB [18]. Los filtros pasa baja son filtros sencillos tipo RC, con 6 dB/octava[3], o lo que es lo mismo, -20 dB/década[4], de *roll-off*. Una constante de tiempo de 1 segundo se refiere a un filtro cuyo punto de -3 dB tiene lugar a 0.16 Hz, y que tiene una pendiente de *roll-off* de 6 dB/octava por encima de los 0.16 Hz. Habitualmente hay dos filtros sucesivos de tal forma que el filtro conjunto disponga de un *roll-off* de, o bien 6 dB, o bien a 12 dB por octava. La constante de tiempo está referida al punto de -3 dB de cada filtro individual (no al filtro conjunto).

La noción de constante de tiempo proviene del hecho de que se supone que la salida propiamente dicha es una señal en DC. De hecho, cuando existe ruido en la entrada, aparece ruido en la salida. Mediante el incremento de la constante de tiempo, la salida se vuelve más estable y es más sencillo realizar la medición de forma fiable.

Conviene advertir que el SR850 muestra la constante de tiempo y el ancho de banda equivalente de ruido (ENBW, *equivalent noise bandwidth*) en el menú de constante de tiempo. El ENBW NO es el polo a -3 dB del filtro, es el ancho de banda efectivo para el ruido gaussiano. Comentaremos más acerca de esto posteriormente.

Filtros analógicos vs filtros digitales

El SR850 mejora en muchos aspectos los filtros analógicos. En primer lugar, los lock-in analógicos proporcionan, como mucho, dos etapas de filtrado con un *roll-off* de 12 dB/octava. Esta limitación generalmente puede achacarse al espacio y a los costes. Cada filtro necesita de varios ajustes o parámetros diferentes de la constante de tiempo. Los diversos parámetros necesitan varios componentes y botones para seleccionarlos, lo cual es costoso y además requiere espacio.

El procesador digital de señal en el SR850 controla todo el filtrado pasa baja. Se puede acoplar a la salida de cada PSD un máximo de cuatro etapas de filtrado para tener un total de 24 dB/octava de *roll-off*. Ya que los filtros son digitales, el SR850 no está limitado a únicamente dos etapas de filtrado.

¿Por qué es deseable un incremento del *roll-off*? Considérese un ejemplo en el que la referencia está ajustada a 1 kHz y existe una fuerte señal de ruido a 1.05 kHz. Este ruido a la salida del PSD es de 50 Hz (diferencia) y 2.05 kHz (suma). Resulta evidente que es más difícil realizar el filtrado pasa baja de la componente de 50 Hz. Si la señal de ruido está 80 dB por encima de la señal a fondo de escala[5] y quisiéramos medir la señal al 1 % (-40 dB), entonces la componente de 50 Hz necesita ser reducida 120 dB. Realizar esto en dos etapas requeriría una constante de tiempo de al menos tres segundos. Llevar a cabo la misma atenuación en cuatro etapas requiere una constante de tiempo de tan sólo 100 ms. En el segundo caso, la respuesta de la salida será 30 veces más rápida y el experimento llevará menos tiempo.

[3] Una `octava` es la distancia que hay entre una frecuencia f y la frecuencia $2 \times f$. Ver [59], sección "Curvas de respuesta de un filtro".

[4] De forma similar al concepto de `octava`, una `década` es la distancia que hay entre una frecuencia f y la frecuencia $10 \times f$. Ver [59], sección "Curvas de respuesta de un filtro".

[5] Una `señal a fondo de escala` es aquella señal que ocupa la escala completa, es decir, que el máximo de la señal es el máximo de la escala, y lo mismo para el mínimo.

Filtros síncronos

Otra ventaja del filtrado digital es la habilidad de realizar filtrado síncrono. Incluso si la señal de entrada está libre de ruido, la salida del PSD siempre contiene una componente a 2F (suma de frecuencias de la señal y la referencia) cuya amplitud iguala o supera la salida en DC deseada dependiendo de la fase. A frecuencias bajas, la constante de tiempo necesaria para atenuar la componente 2F puede ser bastante grande. Por ejemplo, a 1 Hz, la salida 2F está a 2 Hz; y atenuar 6 dB esa señal de 2Hz en dos etapas requiere una constante de tiempo de 3 segundos.

En cambio, un filtro síncrono opera de forma totalmente diferente. La salida del PSD se promedia durante un ciclo completo de la frecuencia de referencia. El resultado es que todas las componentes que sean múltiplos de la referencia (incluyendo 2F) quedan completamente eliminadas. En el caso de una señal pura no sería necesario casi ningún filtrado adicional. Esto resulta cada vez más útil a medida que la frecuencia de referencia se hace más baja.

En el SR850, los filtros síncronos están disponibles a frecuencias de detección inferiores a 200 Hz. A frecuencias mayores, los filtros son innecesarios (2F es fácil de eliminar sin ser necesario el uso de grandes constantes de tiempo). Por debajo de 200 Hz, la salida del filtro síncrono pasa por dos etapas de filtrado normales consecutivas. Esta combinación de filtros elimina todos los múltiplos de la frecuencia de referencia y también proporciona una atenuación de ruido global.

Constantes de tiempo elevadas

Si la frecuencia de referencia es inferior 1 Hz y existe gran cantidad de ruido de baja frecuencia, la salida del PSD contiene muchas componentes de muy baja frecuencia. Sin embargo, no es sencillo trabajar con constantes de tiempo superiores a 100 segundos utilizando filtros analógicos, ya que el condensador necesario para el filtro RC es prohibitivamente grande. El SR850 puede ofrecer constantes de tiempo de hasta 30000 segundos a frecuencias de referencia inferiores a los 200 Hz. Evidentemente, las constantes de tiempo tan elevadas no se utilizan a menos que sean absolutamente necesarias, pero están disponibles.

Ganancia de la salida en DC

El valor de la salida en DC del PSD depende de la reserva dinámica. Con 60 dB de reserva dinámica una señal de ruido puede ser 1000 veces (60 dB) mayor que una señal a fondo de escala. Para el PSD, el ruido no puede superar su rango de entrada. En un lock-in analógico, el rango de entrada del PSD podría ser de 5V. Con 60 dB de reserva dinámica, la señal será de tan sólo 5 mV a la salida del PSD. Generalmente el PSD no tiene ganancia, por lo que la salida en DC del PSD sería de tan sólo unos pocos milivoltios. Incluso si el PSD no tuviera errores en la salida en DC, la tarea de amplificar esta señal de milivoltios hasta los 10 V es propensa a errores. Es necesario que la ganancia de salida en DC sea aproximadamente la misma que la reserva dinámica (1000 en este caso) para obtener una salida de 10 V para una señal de entrada a fondo de escala. Un offset tan pequeño como 1 mV aparecerá como 1 V en la salida. De hecho, el OFFSET de salida del PSD más el offset de entrada del amplificador de DC necesita ser del orden de 10 μV para no afectar a la medida. Si la reserva dinámica se incrementa hasta los 80 dB, entonces este offset todavía necesita ser unas 10 veces menor. Esta es una de las razones por las que los lock-in analógicos no responden bien cuando trabajan con reservas dinámicas grandes.

El lock-in digital no tiene un amplificador de DC analógico. La ganancia de salida sigue siendo otra función controlada por el procesador digital de señal. Ya sabíamos que el PSD digital no tiene

offset de salida en DC. De la misma manera, el amplificador digital en DC no tiene offset de entrada. La amplificación consiste simplemente en tomar los números de entrada y multiplicarles por la ganancia. Esto permite al SR850 operar con una reserva dinámica de 100 dB sin ningún offset de salida.

Resolución

Al igual que en el lock-in analógico, donde el ruido no puede exceder el rango de entrada del PSD, en el lock-in digital el ruido no puede superar el rango de entrada del convertidor analógico digital. Con un conversor A/D de 16 bits, una reserva dinámica de 60 dB significa que, mientras que el ruido tiene de rango los 16 bits completos, la señal a fondo de escala utiliza sólo 6 bits. Con una reserva dinámica de 80 dB, la señal a fondo de escala emplea sólo 2.5 bits. Y con 100 dB de reserva dinámica, la señal es de menos de un bit. Evidentemente multiplicar estos números por una ganancia elevada no va a producir una salida medible. Para conseguir resolución en la salida es necesario recurrir al filtrado. Los filtros pasa baja combinan de forma efectiva muchas muestras de datos juntas. Por ejemplo, con una constante de tiempo de 1 segundo, la salida es el resultado de promediar los datos de los 4 o 5 segundos previos. A una tasa de muestreo de 256 kHz, esto significa que cada dato de salida es el promedio exponencial de los datos de más de un millón de puntos. (Se calcula un nuevo punto de salida cada 4 μs y es un promedio móvil exponencial). ¿Qué sucede cuando se promedian un millón de puntos? En primer lugar, el resultado de promediar tiene un factor de resolución, con respecto a los puntos de entrada, de un millón. Esto representa una ganancia de 20 bits en resolución sobre los datos iniciales. Por consiguiente con elevadas reservas dinámicas (ganancias en DC altas), es necesario introducir filtrado. Las constantes de tiempo más pequeñas no están disponibles cuando la reserva dinámica es muy alta. Esto no es realmente una limitación dado que probablemente exista ruido que haga necesaria una reserva dinámica elevada y por tanto seguirá siendo necesario un filtrado de salida considerable.

2.2.7. Salidas en DC y escalado

El SR850 tiene las salidas X e Y en el panel trasero y las también salidas Canal 1 y 2 (CH1 y CH2) en el panel frontal.

X e Y

Las salidas del panel trasero X e Y son las salidas de los dos detectores sensibles a la fase con filtrado pasa baja, desplazamiento (*offset*) y expansión (*expand*). Estas salidas son las salidas tradicionales de un lock-in analógico. Las salidas X e Y tienen un ancho de banda de salida de 100 kHz.

CH1 y CH2

Las dos salidas del panel frontal pueden ser configuradas para proporcionar proporcionales a X, Y, R, θ, o Trazas 1-4 (Traces 1-4).

Si las salidas se seleccionan como X o Y, estas salidas duplican a sus homólogas del panel trasero.

Si se seleccionan como R o θ, los voltajes de salida son proporcionales a los valores calculados de R y θ. Estos cálculos se efectúan a una frecuencia de 512 Hz y las salidas R y θ se actualizan también a esa misma frecuencia.

Escalas de salida de X, Y, R y θ

La sensibilidad del lock-in es la amplitud RMS de un seno de entrada (a la frecuencia de referencia) que da lugar a una salida en DC a fondo de escala. Tradicionalmente, fondo de escala significa 10 V DC en las salidas BNC *X*, *Y* o *R*. La ganancia global (desde la entrada hasta la salida) del amplificador es, entonces, 10 V/sensibilidad. Esta ganancia se distribuye entre la ganancia en AC previa al PSD y la ganancia en DC posterior al PSD. La modificación de la reserva dinámica a una sensibilidad dada varía la distribución de la ganancia, manteniendo inalterada la ganancia global.

El SR850 considera 10 V como fondo de escala para cualquier salida proporcional únicamente a X, Y o R. Esta es la escala o rango de salida para las salidas X e Y del panel trasero, así como las de las salidas CH1 y CH2 cuando se configuran como salidas de X, Y o R. Cuando las salidas CH1 o CH2 son proporcionales a una traza de datos que está definida simplemente como Y, Y o R, el rango de salida son los 10 V del fondo de escala.

Los amplificadores lock-in están diseñados para medir el valor RMS de la señal de entrada en AC. Todas las sensibilidades y las salidas *X*, *Y* y *R*, y su visualización en pantalla son valores RMS.

La fase es una cantidad que varía de -180º a +180 independientemente de la sensibilidad. Cuando aparece por las salidas CH1 o CH2 un voltaje proporcional a θ, la escala o rango de salida es 18º/Voltio, o lo que es lo mismo, 180º = 10 V.

Desplazamiento (*offset*) y Expansión (*expand*) de las salidas X, Y y R

El SR850 tiene la posibilidad de añadir un desplazamiento (*offset*) a las salidas X, Y y R. Esto resulta útil al medir desviaciones de la señal en torno a cierto valor nominal. El desplazamiento puede ser ajustado de tal forma que la salida tenga un desplazamiento cero. Los cambios en la salida pueden entonces leerse directamente desde la pantalla o a partir de los voltajes de salida. El desplazamiento se especifica como un porcentaje del fondo de escala y este porcentaje no cambia cuando la sensibilidad es modificada. Se pueden programar desplazamientos de hasta el $\pm 100\,\%$.

Las salidas X, Y y R pueden también ser expandidas (*expand*). Esta operación simplemente toma la salida (menos su desplazamiento) y lo multiplica por un factor de expansión. De este modo, una señal cuyos valores están en el 10 % del fondo de escala pueden expandirse para dar 10 V de salida en vez de únicamente 1 V. El uso habitual de la expansión es ampliar la resolución de medición en torno a algún valor distinto de cero. Por ejemplo, suponga que una señal tiene un valor nominal de 0.9 mV y se desean medir variaciones pequeñas, por ejemplo de 10 μV, en la señal. La sensibilidad del lock-in necesita ser de 1mV para adecuarse a la señal nominal. Si el desplazamiento se ajusta como el 90 % del fondo de escala, entonces la señal nominal de 0.9 mV dará lugar a una salida de 0 V. Las variaciones de 10 μV en la señal únicamente darán lugar a 100 mV de salida en DC. Si la salida es expandida por 10, estas pequeñas variaciones son aumentadas 10 veces, y dan valores de salida de 1 V en DC.

El SR850 puede expandir la salida por un factor de 1 a 256 siempre y cuando la salida expandida no supere el fondo de escala. En el ejemplo anterior, las variaciones de 10 μV pueden ser expandidas hasta 100 veces antes de que superen el fondo de escala (a una sensibilidad de 1 mV).

La salida analógica con desplazamiento y expansión es

$$\text{Salida} = \left(\frac{\text{Señal}}{\text{Sensibilidad}} - \text{Desplazamiento} \right) \times \text{Expansión} \times 10\ \text{V} \tag{2.9}$$

donde el desplazamiento es un valor inferior a 1 (50 % = 0.5), la expansión es un entero de 1 a 256 y

la salida no puede superar los 10 V. Para el ejemplo anterior,

$$\text{Salida} = \left(\frac{0{,}91\text{ mV}}{1\text{ mV}} - 0{,}9\right) \times 10 \times 10\text{ V} = 1\text{ V} \tag{2.10}$$

para una señal que es 10 μV mayor que los 0.9 mV nominales. (Desplazamiento = 0.9 y expansión = 10).

Las funciones de desplazamiento y expansión de X e Y en el SR850 son funciones de salida, no afectan el cálculo de R o θ. R tiene su propio desplazamiento y expansión de salida.

Visualización de trazas

En la pantalla sólo pueden mostrarse trazas de datos. Para mostrar el valor X, es necesario definir una traza como X y una vez hecho seleccionar la visualización de esa traza.

Los desplazamientos de salida quedan reflejados al visualizar las trazas de datos que dependen de X, Y o R. Por ejemplo, una traza definida como X se ve afectada por el desplazamiento de X. Cuando el desplazamiento de la salida X se ajusta a cero, el valor de la traza mostrado en la pantalla también caerá a cero. Cualquier pantalla que esté mostrando una traza que esté siendo afectada por un desplazamiento no nulo mostrará iluminado un indicador **Offst** en la zona inferior izquierda de la pantalla.

Las expansiones de salida no incrementan los valores mostrados de X, Y o R en las trazas de datos. La expansión incrementa la resolución del valor X, Y o R utilizado para calcular el valor de la traza. Por ejemplo, una traza que esté definida como X no incrementa su valor mostrado cuando X es expandida. Esto se debe a que la función de expansión incrementa la resolución con la que la señal se mide, no el tamaño de la señal de entrada. El valor visualizado se mostrará con mayor resolución pero seguirá siendo visualizado como el valor original de X menos el desplazamiento de X. Cualquier pantalla que esté mostrando una traza que quede afectada por una expansión no unitaria mostrará iluminado un indicador **Expd** en la zona inferior izquierda de la pantalla.

Las trazas de datos complejas se muestran y almacenan en las mismas unidades que la cantidad calculada. Por ejemplo, si se define una traza como $X \times \frac{\theta}{\text{Aux 1}}$, y $X = 1$ mV, $\theta = 37°$ y Aux 1 = 2.34 V, entonces el valor de la traza será $0{,}001 \times \frac{37}{2{,}34}$ Voltios $\cdot$ Grados por Voltio, o lo que es lo mismo, 0.01581 V $\times \frac{\text{Grados}}{\text{V}}$. Este valor queda modificado por la sensibilidad (X es la señal de entrada, no el voltaje de salida) o por la expansión de X. Un desplazamiento de X cambiará, sin embargo, el valor de esta traza.

Escalado de salida de la traza

¿Qué sucede con las salidas CH1 o CH2 proporcionales a trazas de datos que no son simplemente X, Y, R o θ? Si se define una traza como $A \times \frac{B}{C}$, entonces el voltaje de salida de la traza depende de los valores de cada parámetro. Los voltajes de salida de la traza se calculan determinando los voltajes de salida para cada número A, B y C. Los voltajes de salida individuales (-10 V a +10 V) son entonces combinados utilizando la definición de la traza para determinar el voltaje de salida de la traza.

Por ejemplo, suponga que se define una traza como $\frac{X}{R}$. Los parámetros X y R se escalan a 10 V para una señal de entrada a fondo de escala. Si la sensibilidad es 1 V y los valores medidos son X = 500 mV y R = 1 V, la salida X sería 5 V y la salida R sería 10 V. El voltaje de salida de la traza sería simplemente $\frac{X}{R} = \frac{5}{10} = 0{,}5$ V.

Los voltajes de salida para las trazas que sean definidas como A, B, $\frac{A}{C}$, $\frac{B}{C}$, o $A \times \frac{B}{C}$ se calculan utilizando los voltajes de salida de A, B y C. Las trazas definidas como $A \times B$ (es decir, cuando

$A, B \neq 1$ y $C = 1$ en la ecuación general $A \times \frac{B}{C}$) tienen voltajes de salida que son el producto de los voltajes de salida de A y B divididos entre 10.

Por ejemplo, supongamos que una traza se define como $X \times \theta$. El parámetro X varía hasta 10 V para una señal de entrada a fondo de escala y θ lo hace hasta 10 V para un desfase de 180º. Si el valor medido de X es 1 V con una sensibilidad de 1 V, X sería del 100% con respecto al fondo de escala o, lo que es lo mismo, 10 V. Si el desfase es de 180º, entonces θ sería también 10 V. El voltaje de salida de la traza sería, entonces, $(X = 10\text{ V}) \times \frac{(\theta = 10\text{ V})}{10} = 10\text{ V}$. El factor extra con valor 10 permite la exportación del producto de dos valores a fondo de escala.

Los desplazamientos de las salidas X, Y y R quedan reflejados en las salidas de las trazas que dependen de X, Y o R. Por ejemplo una traza definida como X y exportada a través de CH1 o CH2 queda afectada por el desplazamiento de X. Cuando el desplazamiento de la salida X se ajusta a cero, el voltaje de salida de la traza también caerá a cero.

Las expansiones de salida incrementan el voltaje de salida de X, Y o R en las salidas de las trazas. La expansión incrementa los voltajes de salida de X, Y o R en los cálculos de salida de las trazas. Por ejemplo, una traza que sea definida como X y exportada a través de CH1 o CH2 multiplica su voltaje de salida por el factor de expansión cuando X se expande. Esto se debe a que el voltaje de salida de X es expandido.

El rango de voltajes de salida para cada cantidad individual se muestra en la Tabla 2.1.

X, Y, R	$\left(\frac{\text{Señal}}{\text{Sensibilidad}} - \text{Desplazamiento}\right) \times \text{Expansion} \times 10\text{ V}$
θ	$\frac{10\text{ V}}{180°}$
X_n, Y_n, R_n	$\left(\frac{\text{Señal de ruido}}{\text{Sensibilidad}} - \text{Desplazamiento}\right) \times \text{Expansion} \times 10\text{ V}$
Entradas auxiliares 1-4 (Aux In 1-4)	Voltaje de salida = Voltaje de entrada auxiliar
1	1 V
F	5 V - 10 V para cada octava en frecuencia. Por ejemplo, • 1000 Hz = 5 V • 1200 Hz = 6 V • 1600 Hz = 8 V • 1800 Hz = 9 V • 1990 Hz = 9.95 V • 2000 Hz = 5 V (siguiente octava) Las octavas se definen como sigue: • … • 62.5 Hz - 125 Hz • 125 Hz - 250 Hz • 250 Hz - 500 Hz • 500 Hz - 1000 Hz • 1 kHz - 2 kHz • 2 kHz - 4 kHz • 4 kHz - 8 kHz • 8 kHz - 16 kHz • …

Cuadro 2.1: Tabla de voltajes de salida para cada variable obtenida por el Lock-In

2.2.8. Reserva dinámica

A continuación se clarifica el concepto de reserva dinámica, que ha sido mencionado en diversas ocasiones.

Supongamos que la entrada del lock-in consiste en una señal a fondo de escala con frecuencia f_{REF} mas ruido a otra frecuencia diferente. La definición tradicional de reserva dinámica es el cociente de la mayor señal de ruido tolerable respecto a la señal a fondo de escala, expresado en dB. Por ejemplo, si el fondo de escala es de 1 μV, entonces una reserva dinámica de 60 dB significa que se pude tolerar en la entrada ruido tan grande como 1 mV (60 dB mayor que el fondo de escala) sin sobrecarga.

El problema con esta definición es la palabra tolerable. Evidentemente el ruido en el límite de reserva dinámica no debería causar una sobrecarga en cualquier parte del instrumento —ni en el amplificador de señal a la entrada, ni en el PSD, ni en el filtro pasa baja ni en el amplificador DC. Esto se consigue ajustando la distribución de la ganancia. Para lograr una alta reserva, la ganancia de la señal de entrada se ajusta lo suficientemente baja como para que resulte improbable que el ruido provoque sobrecarga. Esto significa que la señal en el PSD también es muy baja. El filtro pasa baja elimina entonces las componentes mayores de la señal de ruido a la salida del PSD, lo que permite al componente en DC restante ser amplificado (mucho) hasta alcanzar los 10 V de fondo de escala. No hay problema alguno en hacer funcionar el amplificador de entrada a bajas ganancias. Sin embargo, tal y como hemos discutido previamente, los lock-ins analógicos tienen problemas con las reservas elevadas debido a la linealidad del PSD y del amplificador DC. En un lock-in analógico, las grandes señales de ruido casi siempre afectan a la medición de alguna manera.

El problema más común es un error de salida en DC provocado por la señal de ruido. Éste puede aparecer como un offset o como un error de ganancia. Dado que ambos efectos dependen de la amplitud y de la frecuencia del ruido, no pueden ser desplazados (offset) a cero en todos los casos y limitarán la precisión de la medición. Dado que los errores tienen naturaleza continua (son en DC), incrementar la constante de tiempo no resulta de utilidad. La mayoría de los lock-in definen el ruido tolerable como los niveles de ruido que no afectan a la salida en más que un pequeño porcentaje del fondo de escala. Esto es más estricto que simplemente 'no sobrecargar'.

Otro efecto de una reserva dinámica alta es la generación de ruido y deriva a la salida. Esto sucede debido a que el amplificador de salida en DC está funcionando a ganancias muy altas y el ruido de baja frecuencia y la deriva del desplazamiento a la salida del PSD o la entrada del amplificador en DC se verán amplificados y aparecerán mayores a la salida. Es más tolerable el ruido que los errores de deriva en DC, ya que el incremento de la constante de tiempo atenuará el ruido. La deriva en DC en un lock-in analógico es, a menudo, del orden de 1000 ppm/ºC cuando se utiliza una reserva dinámica de 60 dB. Esto significa que el cero se mueve en un 1 % del fondo de escala cuando hay un cambio de 10º C en la temperatura. Es a esto a lo que se considera, generalmente, como el límite de lo tolerable.

Recientemente, la reserva dinámica depende de la frecuencia del ruido. Es evidente que el ruido a la frecuencia de referencia llegará hasta la salida sin atenuación. De modo que la reserva dinámica a la frecuencia de referencia f_{REF} es de 0 dB. A medida que la frecuencia del ruido se aleja de la frecuencia de referencia, la reserva dinámica se incrementa. Esto se debe a que el filtro pasa baja que hay después del PSD atenúa los componentes de la señal de ruido. Tengamos en cuenta que las salidas del PSD están a una frecuencia de $|f_{NOISE} - f_{REF}|$. La tasa a la que se incrementa la reserva dinámica depende del y de la constante de tiempo del filtro pasa baja. La reserva se incrementa al mismo ritmo que la pendiente de *roll off* del filtro. Este es el motivo por el que los filtros de 24 dB/octava son mejores que los de 6 o 12 dB/octava. Cuando la frecuencia del ruido está muy alejada, la reserva está limitada por la distribución de la ganancia y el nivel de sobrecarga de cada elemento de ganancia. El nivel de reserva es la reserva dinámica a la que están referidas las especificaciones.

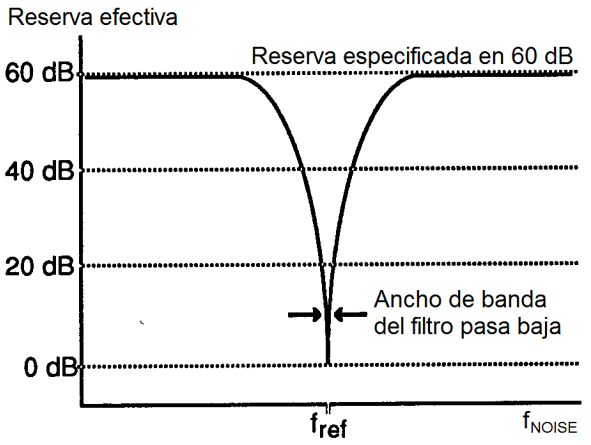


Figura 2.6: Relación entre la reserva efectiva y la frecuencia de la señal de ruido.

El gráfico superior muestra la reserva efectiva[6] en comparación con la frecuencia del ruido. En algunos instrumentos, la entrada de señal atenúa frecuencias alejadas del rango de operación del Lock-in ($f_{NOISE} \gg 100$ kHz). En estos casos la reserva puede ser mayor a esas frecuencias que en el rango de operación. Aunque esta podría ser una buena especificación, la supresión del ruido a frecuencias muy alejadas de la referencia no es algo que necesite de un amplificador lock-in. Los lock-in se utilizan cuando existe ruido a frecuencias cercanas a la señal. Por tanto, la reserva dinámica cuando existe ruido en el rango de operación es más importante.

La reserva dinámica en el SR850

El SR850, con sus detectores digitales sensibles a la fase, no sufre de errores de salida en DC provocados por señales de ruido fuertes. La reserva dinámica puede incrementarse a más de 100 dB sin dar lugar a errores en la medición. Las señales de ruido fuertes no provocan errores de salida en el PSD. La ganancia en DC elevada no da lugar a un incremento en la deriva de salida.

De hecho, la única desventaja de utilizar reservas dinámicas ultra altas ($>$ 60 dB) es el cada vez mayor ruido de salida, debido al ruido en el conversor analógico digital. Este incremento en el ruido de salida únicamente está presente cuando se cumple tanto que la reserva dinámica se incrementa por encima de los 60 dB como que la reserva dinámica está por encima de la reserva mínima. (Si la reserva mínima es de 80 dB, subir a 90 dB podría incrementar el ruido. Tal y como discutiremos a continuación, la reserva mínima no implica un incremento en el ruido de salida, no importa cuán grande sea.)

Para ajustar una escala, el ruido de salida del SR850 con una reserva dinámica de 100 dB sólo pude medirse cuando la señal de entrada está conectada a tierra. A modo de ejemplo, si la referencia del lock-in está a 1 kHz y se aplica una señal grande a 9.5 kHz. Si la señal se incrementa hasta el límite de la reserva dinámica (100 dB mayor que el fondo de escala), la salida mostrará el ruido de la señal a 1kHz. El espectro de cualquier generador de ondas sinusoidales puras siempre tiene ruido de fondo, esto es, existe algo de ruido a cada todas las frecuencias. De modo que, aunque la señal aplicada está a 9.5 kHz, existirá ruido a todas las demás frecuencias, incluida la de la referencia del lock-in a 1 kHz. Este ruido será detectado por el lock-in y aparecerá como ruido a la salida. Este ruido de salida

[6] También puede traducirse como reserva real.

será típicamente mayor que el ruido de salida del propio SR850. De hecho, prácticamente todas las fuentes de señal tendrán un ruido de fondo que será predominante en el ruido de salida del lock-in. Naturalmente, las señales de ruido son generalmente mucho más ruidosas que los generadores de señales puras y tendrán fondos de ruido con un ancho de banda mucho mayor.

Si el ruido no alcanza el límite de la reserva, el ruido de salida del propio SR850 podría volverse detectable a reservas ultra altas. En este caso, simplemente debe reducirse la reserva dinámica y la ganancia en DC bajará y el ruido de salida también bajará. No debe trabajarse con más reserva de la necesaria. No deben utilizarse reservas ultra altas cuando no hay prácticamente ruido.

La dependencia de la frecuencia de la reserva dinámica es inherente a la técnica de detección del lock-in. El SR850, al disponer de más etapas de filtrado pasa baja, puede incrementar la reserva dinámica cerca de la frecuencia de referencia. La reserva especificada se aplica a las señales de ruido en el rango operativo del lock-in, esto es, frecuencias inferiores a 100 kHz. La reserva a frecuencias mayores es de hecho mayor, pero generalmente no es tan útil.

Reserva dinámica mínima

El SR850 siempre dispone de una cantidad mínima de reserva dinámica. Esta reserva mínima cambia con la sensibilidad (ganancia) del instrumento. A ganancias elevadas (sensibilidad a fondo de escala de 50 μV o inferior), la reserva dinámica mínima se incrementa a partir de los 37 dB al mismo ritmo al que la sensibilidad se incrementa. Por ejemplo, la reserva mínima a una sensibilidad de 5 μV es de 57 dB. En muchos Lock-in analógicos la reserva puede ser inferior, pero éste no es el caso para el SR850.

En efecto, en un Lock-in analógico, una reserva inferior significa menor error de salida y deriva. En el SR850, una reserva mayor no incrementa el error de salida o la deriva. Sin embargo, una reserva mayor puede incrementar el error de salida. No obstante, si la ganancia de la señal analógica previa al conversor A/D es lo suficientemente alta, el ruido de 5 $\frac{\text{nV}}{\sqrt{\text{Hz}}}$ de la señal de entrada quedará amplificado a un nivel mayor que el ruido de entrada del conversor A/D. En este punto, el ruido detectado reflejará el propio ruido de la señal de entrada y no el ruido introducido por el conversor A/D. El hecho de incrementar la ganancia analógica (reduciendo la reserva) no reducirá el ruido de salida. Por tanto, no hay razón alguna para reducir la reserva.

A una sensibilidad de 5 μV, la ganancia analógica es lo suficientemente alta como para que el ruido del conversor A/D no suponga un problema. Las sensibilidades por debajo de los 5 μV no requieren más ganancia dado que no mejorará la relación señal ruido (predomina el ruido del front-end). El SR850 no incrementa la ganancia por debajo de 5 μV de sensibilidad, en lugar de eso, se incrementa la reserva mínima. Evidentemente, se puede reducir la ganancia de entrada e incrementar la reserva, en cuyo caso el ruido del conversor A/D podría ser detectado en ausencia de señal de entrada.

2.2.9. Amplificador y filtros de la entrada de señal

Un lock-in pude medir señales tan pequeñas como de hasta unos pocos nanovoltios. Es necesario un amplificador de señal de bajo ruido para elevar la señal a un nivel en el cual el conversor A/D pueda digitalizar la señal sin degradar la señal a ruido. La ganancia analógica en el SR850 varía de apenas 7 a 1000. Como se comentó anteriormente, las ganancias mayores no mejoran la relación señal ruido y no son necesarias.

La ganancia global (AC más DC) está determinada por la sensibilidad. La distribución de la ganancia (AC versus DC) viene dada por la reserva dinámica.

Ruido de entrada

El ruido de entrada del amplificador de señal del SR850 está en torno a 5 $\frac{\text{nVrms}}{\sqrt{\text{Hz}}}$. ¿Qué significa este factor de ruido[7]? Si un amplificador tiene 5 $\frac{\text{nVrms}}{\sqrt{\text{Hz}}}$ de ruido de entrada de ruido de entrada y una ganancia de 1000, entonces la salida tendrá 5 $\frac{\mu\text{Vrms}}{\sqrt{\text{Hz}}}$ de ruido. Supongamos que la salida del amplificador sufre un filtrado pasa baja con un único filtro RC (6 dB/octava de *roll off*) con una constante de tiempo de 100 ms. ¿Cuál será el ruido a la salida del filtro?

El ruido de entrada del amplificador y el ruido Johnson[8] de las resistencias son de naturaleza gaussiana. Esto significa que la cantidad de ruido es proporcional a la raíz cuadrada del ancho de banda en el que se mide el ruido. Un filtro RC de una sola etapa tiene un ENBW de $\frac{1}{4}$ T donde T es la constante de tiempo ($R \times C$). Esto significa que el ruido gaussiano existente a la entrada del filtro queda eliminado con un ancho de banda efectivo igual al ENBW. En este ejemplo, el filtro ve 5 $\frac{\mu\text{Vrms}}{\sqrt{\text{Hz}}}$ de ruido a su entrada. Tiene un ENBW de $\frac{1}{4\times 100\text{ms}}$ o 2.5 Hz. El ruido en la señal de la tensión a la salida del filtro será de 5 $\frac{\mu\text{Vrms}}{\sqrt{\text{Hz}}} \times \sqrt{2{,}5}$ Hz o 7.9 μVrms. Para el ruido gaussiano el ruido pico a pico es aproximadamente 5 veces superior al ruido RMS. Por tanto, la salida tendrá aproximadamente 40 μV pico a pico de ruido.

El ruido de entrada funciona de la misma manera para un lock-in. Para aquellas sensibilidades inferiores a valores en torno a 5 μV a fondo de escala, el ruido de entrada determinará el ruido de salida (empleando una reserva mínima). La cantidad de ruido a la salida queda determinado por el ENBW del filtro pasa baja. El SR850 muestra el ENBW en el menú de constante de tiempo (Time Constant). El ENBW depende de la constante de tiempo y del *roll off* del filtro. Por ejemplo, suponga que el SR850 se ajusta a 5 μV a fondo de escala con una constante de tiempo de 100 ms y 6 dB/octava de *roll off* del filtro. El lock-in medirá el ruido de entrada con un ENBW de 2.5 Hz. Esto se traduce en 7.9 nVrms a la entrada. En la salida, esto representa aproximadamente el 0.16 % del fondo de escala ($\frac{7{,}9\text{ nV}}{5\mu\text{V}}$). El ruido pico a pico estará en torno al 0.8 % del fondo de escala.

Todo lo anterior asume que la entrada de señal proviene de una fuente de baja impedancia. Recuerde que las resistencias tienen un ruido Johnson igual a $0{,}13 \times \sqrt{R}\,\frac{\text{nVrms}}{\sqrt{\text{Hz}}}$. ¡Incluso una resistencia de 50 Ω tiene casi 1 $\frac{\text{nVrms}}{\sqrt{\text{Hz}}}$ de ruido! Una impedancia de la fuente de señal de 2 kΩ tendrá un ruido Johnson superior al ruido de entrada del SR850. Para determinar el ruido conjunto de varias fuentes de ruido, tome la raíz cuadrada de la suma de los cuadrados de los factores de ruido individuales. Por ejemplo, si se emplea una impedancia de fuente de 2 kΩ, el ruido de Johnson será de 5.8 $\frac{\text{nVrms}}{\sqrt{\text{Hz}}}$. El ruido conjunto a la entrada del SR850 será de $\sqrt{5^2 + 5{,}8^2}$ o 7.7 $\frac{\text{nVrms}}{\sqrt{\text{Hz}}}$.

Discutiremos más sobre las fuentes de ruido posteriormente, en esta misma sección.

A ganancias bajas (con sensibilidades por encima de los 50 μV) no hay suficiente ganancia a reserva alta como para amplificar el ruido de entrada a un nivel mayor que el ruido del conversor A/D. En estos casos, el ruido de salida está determinado por el ruido del conversor A/D. Afortunadamente, a estas sensibilidades, la ganancia en DC es baja y el ruido a la salida es casi inexistente.

Filtros rechaza banda

El SR850 tiene dos filtros rechaza banda en la cadena de amplificación de señal. Estos están presintonizados a la frecuencia de la red eléctrica (50 o 60 Hz) y al doble de la frecuencia de la red eléctrica (100 o 120 Hz). En aquellas circunstancias en las que las señales de ruido más fuertes estén a

[7] Ver [45], página 30 (*noise figure*).

[8] Ver [45], página 27.

la frecuencia de la red eléctrica, estos filtros pueden ser activados para eliminar las señales de ruido a dichas frecuencias. El hecho de eliminar las señales de ruido más fuertes antes de la etapa de ganancia final puede reducir la cantidad de reserva dinámica necesaria para realizar una medición. Se podría obtener algo de mejora en la medida en la que estos filtros reducen la reserva necesaria hasta o bien 60 dB, o bien la reserva mínima (cualquiera de las dos es superior). Si, o bien la reserva necesaria sin estos filtros rechaza banda está por debajo de los 60, o bien reserva mínima es suficiente, estos filtros no supondrán una mejora significativa de la medida.

El uso de alguno de estos filtros impide hacer mediciones próximas a las frecuencias que son rechazadas. Estos filtros tienen un rango de atenuación finito, generalmente de aproximadamente 10 Hz. Por tanto, si el lock in está realizando medidas a 70 Hz, ¡no emplee el filtro rechaza banda de 60 Hz! La señal quedará atenuada y la medida será errónea. A la hora de medir desplazamientos de fase, estos filtros pueden afectar las mediciones de fase hasta a una octava de distancia.

Filtro de suavizado

Después de todo el filtrado de la señal y amplificación, hay un filtro de suavizado. El proceso de digitalización de la señal requiere de este filtro. De acuerdo al criterio de Nyquist, las señales deben ser muestreadas a una frecuencia que sea como mínimo el doble de la frecuencia más alta de la señal. En este caso, la frecuencia más alta de la señal es de 100 kHz y la frecuencia de muestreo es de 256 kHz. Sin embargo, no se debe permitir que cualquier señal por encima de 128 kHz alcance el conversor A/D. Estas señales violarían el criterio de Nyquist y serían submuestreadas. El resultado de este submuestreo es que estas señales de mayor frecuencia quedarían como si fueran de frecuencias inferiores en el flujo de datos digital. Por tanto, una señal a 175 kHz aparecería como si estuviera por debajo de 100 kHz en el flujo de datos digital y sería detectable por el PSD digital. Esto sería un problema.

Para evitar este submuestreo, la señal analógica se filtra para eliminar cualquier señal por encima de 154 kHz (al muestrear a 256 kHz, las señales por encima de 154 kHz aparecerían como si estuvieran por debajo de 102 kHz). Este filtro tiene un rango de frecuencia de paso desde DC hasta 102 kHz, de tal manera que no afecte las mediciones en el rango operativo del lock-in. El filtro elimina las señales con frecuencias comprendidas entre 102 kHz y 154 kHz y logra una atenuación por encima de los 154 kHz de al menos 100 dB. Las variaciones en la amplitud y los desfases provocados por este filtro están calibrados de fábrica y no afectan a las mediciones. Este filtro es transparente al usuario.

2.2.10. Conexiones de entrada

Con el fin de obtener la mejor precisión para una medición dada, se debe tener cuidado de minimizar las diversas fuentes de ruido que pueden encontrarse en el laboratorio. Para el ruido intrínseco (ruido de Johnson, ruido rosa[9], o ruido de entrada), el experimento o el detector deben ser diseñados con estas fuentes de ruido en mente. Estas fuentes de ruido están presentes independientemente de las conexiones de entrada. Tanto el efecto de las fuentes de ruido en el laboratorio (tales como motores, generadores de señal, etc.) como el problema de las interconexiones entre el detector y el lock-in pueden minimizarse poniendo especial cuidado en las conexiones de entrada.

Existen dos métodos básicos para conectar una señal de voltaje al lock-in; la que incluye conexión a tierra, o asimétrica, resulta más cómoda, mientras que la conexión diferencial elimina ruidos y señales espurias de manera más efectiva.

[9]El ruido rosa, también llamado ruido 1/f, es aquel cuyo nivel sonoro está caracterizado por una densidad espectral inversamente proporcional a la frecuencia. Ver [34]

Conexión de voltaje asimétrica (A)

En el primer método, el lock-in emplea la entrada A en modo asimétrico (*single-ended*). El lock-in detecta la señal como la diferencia de potencial (el voltaje) entre los conductores central (núcleo) y exterior (apantallado), únicamente, de la entrada A. El lock-in no fuerza la puesta a tierra del apantallado del cable A; en su lugar está internamente conectado a la tierra del lock-in a través de una resistencia. El valor de esta resistencia se elige en el menú INPUT. El modo 'Float' (flotante) utiliza 1 kΩ y el modo 'Ground' (tierra) utiliza 10 Ω. Esto evita problemas de bucle de tierra, entre el experimento y el lock-in, provocados por potenciales de tierra diferentes. El lock-in permite que el apantallado esté 'cuasi-flotante' para poder detectar la tierra del experimento. No obstante, la señal de ruido en el apantallado aparecerá como ruido para el lock-in. Esto representa un problema, dado que el lock-in no puede rechazar este ruido. El ruido de modo común, que aparece tanto en el núcleo como en el apantallado, es rechazado por el CMRR de 100 dB de la entrada del lock-in, pero el ruido que aparece únicamente en el apantallado no es rechazado en absoluto.

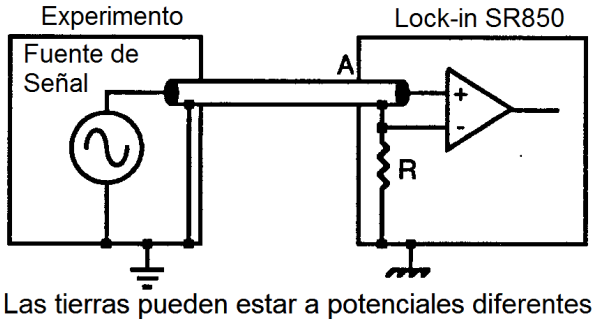


Figura 2.7: Esquema de conexión de entrada en modo asimétrico.

Conexión de voltaje diferencial (*A*-*B*)

El segundo método de conexión es el modo diferencial. El lock-in mide la diferencia de voltaje entre los núcleos de los conductores de las entradas A y B. Ambas conexiones de señal están apantalladas ante las señales espurias.

Cuando se emplean dos cables, es importante que ambos recorran el mismo camino entre el experimento y el lock-in, para evitar problemas de compatibilidad electromagnética.

Señales en modo común

Las señales en modo común son aquellas señales que aparecen iguales, o bien tanto en el núcleo como en el apantallado (A), o bien tanto en A como en B ($A - B$). Es importante, con cualquiera de ambos esquemas de conexión, minimizar tanto el ruido en modo común como la señal en modo común. Conviene darse cuenta de que la fuente de señal se mantiene cercana al potencial de tierra en los esquemas anteriores. El factor o razón de rechazo al modo común (CMRR, *Common Mode Rejection Ratio*) especifica el grado de cancelación. A frecuencias bajas, un CMRR de 100 dB indica que la señal e modo común queda cancelada en una parte de entre 10^5. Incluso con un CMRR de 100 dB, una señal en modo común de 100 mV se comporta como una señal diferencial de 1 μV. Esto resulta especialmente perjudicial si la señal en modo común está a la frecuencia de referencia (lo

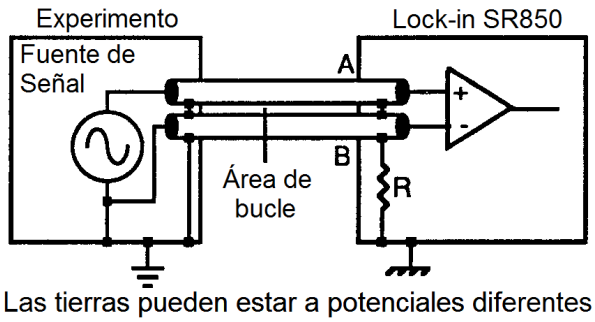


Figura 2.8: Esquema de conexión de entrada en modo diferencial.

cual sucede frecuentemente debido a lazos de tierra, también llamados bucles de masa). El CMRR se reduce en torno a 6 dB/octava (20 dB/década) empezando alrededor de 1 kHz.

Corriente de entrada (*I*)

La corriente de entrada en el SR850 utiliza el BNC de la entrada A. Se elige si se desea entrada de corriente o de voltaje a través del menú INPUT. La entrada de corriente tiene una impedancia de entrada de 1 kΩ y una ganancia de corriente de, o bien 10^6 o bien 10^8 Voltios/Amperio. Se pueden medir corrientes desde 1 μA hasta $2f$ A a fondo de escala.

La impedancia de la fuente de señal es el factor más importante para tener en cuenta a la hora de decidir entre mediciones de corriente y voltaje.

Para fuentes de alta impedancia, superiores a 1 MΩ (ganancia de 10^6) o a 100 MΩ (ganancia de 10^8) y corrientes pequeñas, hay que utilizar la entrada de corriente. Su relativamente baja impedancia reduce enormemente los errores de fase y de amplitud provocados por la constante de tiempo de la impedancia del cable. La capacidad del cable debería mantenerse baja para minimizar la ganancia del ruido de alta frecuencia del preamplificador de corriente.

Para fuentes con impedancias moderadas a bajas, o corrientes mayores, se prefiere el uso de la entrada de voltaje. Una resistencia de pequeño valor se puede utilizar para derivar (*shunt*) la corriente de la señal y general una señal de voltaje. El lock-in mide entonces el voltaje en los extremos de la resistencia (*shunt*). El valor de la resistencia *shunt* debe seleccionarse adecuadamente para que no afecte a la fuente de corriente, a la vez que se proporciona señal suficiente para que el lock-in la pueda medir.

¿Qué ganancia de corriente debería utilizarse? La ganancia de corriente determina tanto el ruido de la corriente de entrada al lock-in como el ancho de banda de la medición. Las señales muy por encima del ancho de banda de entrada quedan atenuadas 6 dB/octava. El ruido y el ancho de banda están relacionados con la ganancia de acuerdo a la siguiente tabla:

Ganancia	Ruido	Ancho de banda
10^6	$130f \frac{\text{A}}{\sqrt{\text{Hz}}}$	70 kHz
10^8	$13f \frac{\text{A}}{\sqrt{\text{Hz}}}$	700 Hz

Cuadro 2.2: Relación entre el ruido de la corriente de entrada y el ancho de banda con la ganancia seleccionada.

La ganancia de corriente se selecciona en el menú INPUT cuando la entrada I está en uso.

Acoplamiento en AC vs Acoplamiento en DC

La señal de entrada puede estar acoplada tanto en AC como en DC. Se debería utilizar el acoplamiento en AC a frecuencias por encima de los 50 mHz cuando resulte posible. A frecuencias más bajas, el acoplamiento en DC es obligatorio.

Una señal DC, si no queda eliminada por el filtro de acoplamiento en AC, será multiplicada por la onda sinusoidal de referencia y producirá una salida a la frecuencia de referencia. Esta señal no está normalmente presente y necesita ser eliminada por el filtro pasa baja. Si la componente en DC de la señal es elevada, la salida será elevada y será necesaria una constante de tiempo grande para eliminarla. El acoplamiento en AC elimina el componente en DC de la señal sin sacrificar la señal, siempre y cuando la frecuencia esté por encima de 160 mHz.

La corriente de entrada al preamplificador de voltaje siempre está acoplada en DC. Se puede activar el acoplamiento en DC como etapa posterior al preamplificador de corriente para eliminar cualquier señal de corriente en DC.

2.2.11. Fuentes de ruido intrínsecas (aleatorias)

Un buen diseño experimental puede reducir las fuentes de ruido y mejorar la estabilidad y precisión de las mediciones.

Existen varias fuentes de ruido intrínsecas que están presentes en todas las señales electrónicas. Estas fuentes tienen origen físico.

Ruido Johnson (o ruido térmico)

Cada resistencia genera un voltaje de ruido entre sus terminales debido a las fluctuaciones térmicas en la densidad de carga de la propia resistencia[10]. Estas fluctuaciones dan lugar a un voltaje de ruido en circuito abierto,

$$V_{RUIDO}(\text{rms}) = \sqrt{4kTR\Delta f} \tag{2.11}$$

donde k es la constante de Boltzmann ($1,38 \times 10^{-23}\ \frac{\text{J}}{^{\circ}\text{K}}$), T es la temperatura en grados Kelvin (típicamente 300 ºK), R es la resistencia en Ohmios, y Δf es el ancho de banda en Hercios. Δf es el ancho de banda de la medición.

Dado que el amplificador de señal de entrada del SR850 tiene un ancho de banda de aproximadamente 300 kHz, el ruido eficaz a la entrada del amplificador es $V_{RUIDO}(\text{rms}) = 70\sqrt{R}$ nVrms o $350\sqrt{R}$ nV pico a pico. Este ruido es de banda ancha. La cantidad de ruido medida por el lock-in está determinada por el ancho de banda de la medición. Cabe recordar que el lock-in no estrecha su ancho de banda de detección hasta después de los detectores sensibles a la fase. En un lock-in, el ancho de banda equivalente de ruido (ENBW) del filtro pasa baja (la constante de tiempo) es lo que determina el ancho de banda de detección. En este caso, el ruido medido de una resistencia a la entrada del lock-in, típicamente la impedancia de entrada para la señal, es sencillamente

$$V_{RUIDO}(\text{rms}) = 0{,}13\sqrt{R}\sqrt{\text{ENBW}}\,\text{nV} \tag{2.12}$$

El SR850 muestra el ENBW en el menú TIME CONSTANT. Este es el ancho de banda de ruido correcto para la constante de tiempo y el número de polos y debería utilizarse para calcular el ruido de Johnson detectado. El ENBW mostrado no tiene en cuenta al filtro síncrono.

El ancho de banda del amplificador de señal determina la cantidad de ruido de banda ancha que será amplificado. Esto afecta a la reserva dinámica. La constante de tiempo determina la cantidad de ruido que será medida a la frecuencia de referencia.

[10] Ver [65], página 109.

Ruido impulsivo (o ruido de disparo)

La corriente eléctrica tiene ruido debido a la naturaleza finita de los portadores de carga. Existe siempre algo de no uniformidad en el flujo de electrones, lo que genera ruido en la corriente. A este ruido se le llama ruido de disparo[11]. Éste puede aparecer como voltaje de ruido cuando la corriente pasa a través de una resistencia, o como ruido al medir la corriente. El ruido de disparo o ruido de corriente viene dado por la ecuación

$$I_{RUIDO}(\text{rms}) = \sqrt{2qI\Delta f} \tag{2.13}$$

donde q es la carga del electrón ($1{,}6 \times 10^{-19}$ Culombios), I es la corriente RMS en AC o corriente en DC dependiendo del circuito, y Δf es el ancho de banda.

Cuando la corriente de entrada de un lock-in se emplea para medir la corriente de una señal en AC, el ancho de banda es, generalmente, tan pequeño que el ruido de disparo no es relevante.

Ruido rosa

Cada resistencia de 10 Ω, no importa de qué material esté fabricada, tiene el mismo ruido de Johnson. Sin embargo, existe un ruido adicional, además del ruido de Johnson, que proviene de las fluctuaciones en la resistencia debidas a la corriente fluyendo a través de la resistencia. Para las resistencias cuya composición está basada en el carbono, éste está típicamente entre 0.1 μV y 3 μV rms de ruido por voltio aplicado a la resistencia. Las resistencias de película metálica y las resistencias de hilo bobinado tienen en torno a 10 veces menos ruido. Este ruido tiene un espectro de $\frac{1}{f}$ y hace más difíciles las mediciones a baja frecuencia.

Otras fuentes de ruido rosa incluyen el ruido que puede encontrarse en válvulas de vacío y semiconductores.

Ruido total

Todas estas fuentes de ruido son incoherentes. El ruido aleatorio total es la raíz cuadrada de la suma de los cuadrados de todas las fuentes de ruido incoherentes.

2.2.12. Fuentes de ruido externas

Además de las fuentes de ruido intrínsecas explicadas anteriormente, existen diversas fuentes de ruido externas en el interior del laboratorio.

Muchas de estas fuentes de ruido son asíncronas, esto es, no están relacionadas con la referencia y no tienen lugar a la frecuencia de referencia o a sus armónicos. Algunos ejemplos incluyen luminarias, motores, equipos de refrigeración, radios, pantallas de ordenador, etc. Estas fuentes de ruido afectan a la medición obligando al incremento de la reserva dinámica necesaria o aumentando la constante de tiempo.

No obstante, algunas fuentes de ruido guardan relación con la referencia y, si entran a formar parte de la señal, se sumarán o restarán a la auténtica señal y provocarán errores en la medición. Fuentes típicas de ruido síncrono son los bucles de tierra entre el experimento, el detector y el lock-in, y el ruido electrónico del oscilador de referencia o del montaje experimental.

Muchas de estas fuentes de ruido pueden ser minimizadas con buenas prácticas de laboratorio y con un buen diseño del experimento. Existen varias formas mediante las cuales las fuentes de ruido se acoplan en el recorrido de la señal.

[11] Ver [95], página 35.

Acoplamiento capacitivo

Un voltaje en AC proveniente de algún aparato cercano puede acoplarse con un detector mediante capacidades parásitas. A pesar de que $C_{PARASITA}$ puede ser muy pequeña, el ruido acoplado todavía puede ser mayor que una señal débil del experimento. Esto resulta especialmente perjudicial si el ruido acoplado es síncrono (a la frecuencia de referencia).

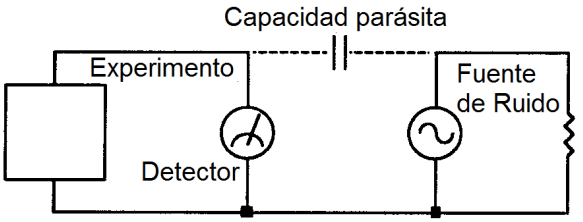


Figura 2.9: Recepción de ruido provocada por el acoplamiento capacitivo (condensador parásito) entre los cables al detector y los de una fuente de ruido próxima.

Podemos estimar la corriente del ruido provocada por una capacidad parásita mediante la fórmula

$$i = C_{PARASITA}\frac{\mathrm{dV}}{\mathrm{dt}} = \omega C_{PARASITA} V_{RUIDO} \tag{2.14}$$

Donde ω es 2π veces la frecuencia del ruido, V_{RUIDO} es la amplitud del ruido, y $C_{PARASITA}$ es la capacidad parásita.

Por ejemplo, si la fuente de ruido es un circuito de alimentación, entonces $f = 60$ Hz y $V_{RUIDO} = 220$ V. $C_{PARASITA}$ puede estimarse utilizando un condensador plano paralelo equivalente. Si la capacidad tiene un área aproximada de 1 cm^2 a una separación de 10 cm, entonces $C_{PARASITA}$ es de 0.009 pF. La corriente de ruido resultante será de 400 pA (a 60 Hz).

Esta pequeña corriente de ruido puede ser cientos de veces mayor que la corriente de la señal. Si la fuente de ruido está a una frecuencia superior, el ruido acoplado será incluso mayor.

Si la fuente de ruido está a la frecuencia de referencia, el problema se agrava. El lock-in rechaza el ruido a cualquier otra frecuencia, pero la recepción a la frecuencia de referencia aparece como si fuera señal.

Algunos remedios para el acoplamiento capacitivo de ruido son los siguientes:

1. Quitar o apagar la fuente de ruido.

2. Mantener la fuente de ruido lejos del experimento (reduciendo $C_{PARASITA}$). No llevar los cables de señal cerca de la fuente de ruido.

3. Diseñar el experimento para medir voltajes con impedancia baja (las corrientes de ruido generan muy poco voltaje).

4. Instalar un apantallamiento capacitivo colocando tanto el experimento como el detector en una caja de metal

Acoplamiento inductivo

Una corriente AC en un aparato cercano puede acoplarse con el experimento a través de un campo magnético. Una corriente variable en un circuito cercano da lugar a un campo magnético variable que induce una f.e.m. ($\frac{d\phi_B}{dt}$) en el lazo que conecta el detector con el experimento. Esto es similar a un transformador en el que el lazo experimento-detector funciona como el devanado secundario.

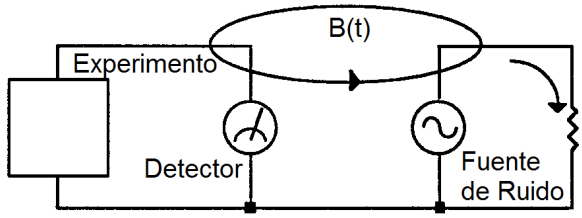


Figura 2.10: Recepción de ruido provocada por el acoplamiento inductivo entre los cables al detector y los de una fuente de ruido próxima.

Algunos remedios para el ruido acoplado por inducción son los siguientes:

1. Quitar o apagar la fuente de ruido que interfiere.
2. Reducir el área del lazo de recepción (o bucle de recepción) utilizando pares trenzados o cables coaxiales, o incluso trenzando los dos cables coaxiales utilizados en conexiones diferenciales.
3. Utilizando apantallamiento magnético para impedir que el campo magnético atraviese el área del experimento.
4. Medir las corrientes, no los voltajes, mediante detectores de alta impedancia.

Acoplamiento resistivo o bucles de tierra

Aquellas corrientes que fluyen a través de las conexiones de tierra pueden dar lugar a voltajes de ruido. Esto resulta un problema especialmente con las corrientes de tierra a la frecuencia de referencia.

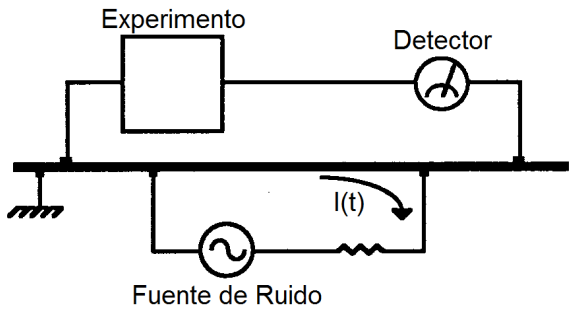


Figura 2.11: Recepción de ruido provocada por un bucle de tierra.

En esta ilustración, el detector está midiendo la señal relativa a una tierra que está ubicada lejos del resto del experimento. El experimento detecta tanto la señal del detector como el voltaje causado por la corriente de retorno de tierra de la fuente de ruido que pasa a través de la resistencia finita de tierra que hay entre el experimento y el detector. El experimento y el detector están conectados a tierra en ubicaciones diferentes, que en este caso, están a potenciales diferentes.

Algunos remedios para los problemas de bucles de tierra incluyen:

1. Que todas las conexiones de tierra estén enganchadas al mismo punto físico.

2. Utilizar una barra de puesta a tierra gruesa para reducir las resistencias de las conexiones a tierra.

3. Eliminar las fuentes de corrientes de tierra grandes de aquellas barras de tierra que sean empleadas para pequeñas señales.

Efectos microfónicos

No todas las fuentes de ruido tienen un origen eléctrico. El ruido mecánico puede traducirse en ruido eléctrico mediante efectos microfónicos. Los cambios físicos en el experimento o en los cables (debidos a vibraciones, por ejemplo) pueden dar lugar a ruido eléctrico en todo el rango de frecuencias del lock-in.

Por ejemplo, consideremos un cable coaxial que esté conectando un detector con un lock-in. La capacidad del cable está en función de su geometría. Las vibraciones mecánicas del cable se traducen en que la capacidad varía en el tiempo, generalmente a la frecuencia de vibración. Dado que el cable está regido por $Q = CV$, tomando derivadas, tenemos que

$$C\frac{\mathrm{d}V}{\mathrm{d}t} + V\frac{\mathrm{d}C}{\mathrm{d}t} = \frac{\mathrm{d}Q}{\mathrm{d}t} = i \tag{2.15}$$

Las vibraciones mecánicas en el cable que provocan un $\frac{\mathrm{d}C}{\mathrm{d}t}$ darán lugar a una corriente en el cable. Esta corriente afecta al detector y a la señal medida.

Algunos métodos para minimizar las señales microfónicas son los siguientes:

1. Eliminar las vibraciones mecánicas próximas al experimento.

2. Atar o asegurar los cables que transmitan señales sensibles de forma que no se muevan.

3. Utilizar un cable de bajo ruido que esté diseñado para reducir los efectos microfónicos.

Efectos termopares

La fuerza electromotriz (f.e.m.) creada por las uniones entre metales diferentes puede comportarse como una fuente de ruido. Esta fuente de ruido es generalmente de baja frecuencia, ya que la temperatura del detector y del experimento, por lo general, varía lentamente. Este efecto suele ser apreciable en el rango de salida de muchos detectores y puede suponer un problema para mediciones de baja frecuencia, en especial para el rango de mHz.

Algunos métodos para minimizar los efectos termopares son:

1. Mantener constante la temperatura del experimento o del detector.

2. Utilizar una unión de compensación, esto es, una segunda unión[12] en polaridad inversa, lo que genera una f.e.m. que cancela el potencial térmico de la primera unión. Esta segunda unión debería mantenerse a la misma temperatura que la primera unión.

2.2.13. Mediciones de ruido

Los amplificadores lock-in pueden utilizarse para medir el ruido. Las mediciones de ruido se emplean generalmente para caracterizar componentes o detectores.

El SR850 mide el ruido de la señal de entrada **a la frecuencia de referencia**. Muchas fuentes de ruido tienen una dependencia de la frecuencia que el lock-in puede medir.

¿Cómo mide el ruido un lock-in?

Recordemos que el lock-in detecta aquellas señales cercanas a la frecuencia de referencia. En concreto, las señales de entrada contenidas en el ancho de banda de detección, marcado por la constante de tiempo y por el *roll off* del filtro pasa baja, aparecen a la salida a la frecuencia $f = f_{SIG} \cdot f_{REF}$. El ruido de entrada próximo a f_{REF} aparece como ruido a la salida con un ancho de banda que va desde DC hasta el ancho de banda de detección.

El ruido puede definirse como la desviación estándar (la raíz cuadrada de la media de las desviaciones al cuadrado) de las mediciones de X, Y o R. El SR850 puede medir este ruido con exactitud, almacenando primero el valor de salida en una tabla para luego calcular la desviación estándar utilizando las funciones matemáticas adecuadas. El ruido, en $\frac{\text{Voltios}}{\sqrt{\text{Hz}}}$, es la desviación estándar dividida por la raíz cuadrada del ancho de banda equivalente de ruido de la constante de tiempo.

Respecto al ruido Gaussiano, el ancho de banda equivalente de ruido (ENBW) de un filtro pasa baja, es el ancho de banda del filtro rectangular perfecto[13] que pasa la misma cantidad de ruido que el filtro real. El ENBW se muestra junto con la constante de tiempo en el menú GAIN/TC.

Estimación de ruido

La técnica anterior, aunque matemáticamente correcta, no puede ofrecer una salida en tiempo real o una salida analógica proporcional al ruido medido. Para estas mediciones, el SR850 puede estimar el ruido de X, Y o R directamente.

Para mostrar o almacenar el ruido de X, por ejemplo, simplemente hay que definir una traza como Xn (en el menú Trace/Scan). El valor de Xn se calcula en tiempo real y es una estimación del ruido de X. Los valores Yn y Rn son estimaciones del ruido de Y y del ruido de R, respectivamente.

El valor Xn se calcula a partir de los valores medidos de X utilizando el algoritmo siguiente. El valor medio actual de X se resta del valor actual de X para hallar la desviación de X de la media. Finalmente, se calcula la media móvil simple del valor absoluto de las desviaciones. A este cálculo se lo denomina la mean average deviation o MAD. El MAD no es lo mismo que el cálculo del RMS. Sin embargo, si el ruido es de naturaleza gaussiana, entonces el ruido RMS y el ruido MAD están relacionados por un factor constante.

El SR850 utiliza el método MAD para estimar los valores de ruido RMS Xn, Yn, y Rn. La ventaja de esta técnica es su simplicidad numérica y su velocidad. Los cálculos de ruido para X, Y y R se realizan a 512 Hz. A cada muestra se calcula la media y la media móvil simple del valor absoluto

[12] Se refiere a uniones entre los mismos dos metales que están provocando el efecto termopar.

[13] A esto se lo conoce como un filtro Sinc. Ver [78], pág 56 según la numeración interna (esquina superior de las páginas), y [19].

de las desviaciones. El tiempo de promediado[14] (para la media y la desviación media) depende de la constante de tiempo. El SR850 selecciona el tiempo de promediado y varía entre 10 veces y 80 veces el valor de la constante de tiempo. Un promediado más corto da lugar a estimaciones de ruido muy pobres (la media varía rápidamente y las desviaciones no son correctamente promediadas). Los tiempos de promediado mayores, aunque proporcionan mejores resultados, tardan más tiempo en dar una respuesta estable.

Para cambiar el tiempo de estabilización hay que cambiar la constante de tiempo. Hay que recordar que tiempos de estabilización más cortos utilizan constantes de tiempo menores (anchos de banda de ruido mayores) y dan lugar a estimaciones de ruido más ruidosas. Los valores Xn, Yn y Rn se muestran en unidades de $\frac{\text{Voltios}}{\sqrt{\text{Hz}}}$. El ENBW de la constante de tiempo ya se tiene en cuenta en el cálculo. Por tanto, el valor medio de Xn no debería depender de la constante de tiempo.

El SR850 realiza el cálculo de ruido constantemente, estén o no siendo mostrados o almacenados los valores Xn, Yn o Rn. En consecuencia, en cuanto se muestra por pantalla Xn, el valor mostrado está actualizado y no se producen retardos. Si se cambia la sensibilidad, la estimación de ruido necesitará un tiempo para asentarse al valor correcto.

Para la mayoría de aplicaciones, la estimación de ruido y la desviación estándar producen la misma respuesta. La decisión de qué método debe utilizarse depende delos requisitos del experimento.

Ruido de R

Supongamos a modo de ejemplo que X e Y son igualmente ruidosas y están centradas en torno al cero. Los valores de R son siempre positivos (magnitud) y por tanto la promediados a un valor distinto de cero. En este caso, el ruido de X e Y da lugar a una media R que puede ser interpretada como el valor mínimo detectable de R. El cálculo del ruido de R por cualquiera de ambos métodos normalmente producirá un valor más pequeño que X y también que Y. Esto se debe a que tanto X como Y tienen valores tanto positivos como negativos centrados en cero, dando lugar a desviaciones grandes, mientras que R siempre es positiva con una media distinta de cero y tiene desviaciones más pequeñas. En este caso, el ruido de R está definido matemáticamente pero no es indicativo del ruido gaussiano que típicamente se mide.

Si el valor de R está en estado estable y distinto de cero, con desviaciones debidas al ruido pequeñas comparadas con la media de R, entonces el valor del ruido de R tiene sentido. Este es el caso cuando se mide ruido en presencia de una señal realmente detectable. En este caso, el valor de Rn se aproxima a los de Xn e Yn.

2.3. Aplicaciones reales

En este apartado se exponen los diversos usos que se le ha dado al amplificador Lock-in. En su mayoría están ligados a técnicas de física, química y electrónica.

Un uso genérico de estos amplificadores se presenta en el artículo de Kolle y O'Leary [71], en el que se muestra el uso de un Lock-in para medir con alta precisión la impedancia de los sensores capacitivos, poniendo como ejemplo un sensor de humedad, y lo implementan utilizando una FPGA, un DAC, un amplificador operacional, y algunos componentes pasivos.

[14] Esto es, la duración de la ventana de tiempo, en la que los datos que están dentro se utilizan para calcular el valor de la media móvil actual.

2.3.1. Balanzas

A la hora de medir pesos y masas, típicamente se han utilizado galgas extensiométricas, que consisten en un circuito en forma de zigzag hecho con un material conductor que se somete a una diferencia de potencial. Gracias al efecto piezorresistivo, la resistencia del material cambia cuando se deforma. Colocando este dispositivo a lo largo de un brazo metálico sujeto por uno de los extremos a una base o plataforma y conectando un platillo encima del extremo libre del brazo, es posible medir la flexión de éste (del brazo) al colocar un objeto encima del platillo. Por tanto, es posible utilizar un amplificador Lock-in para medir la conductancia a través de la galga.

Sin embargo, Abu Al Aish et al. presentaron a principios del año 2010 un modelo de balanza que utilizaba sensores capacitivos en lugar de resistivos [44]. Su diseño consiste en tres cilindros concéntricos de latón. Dos de ellos están anclados a una base de PMMA[15]y están conectados a sendos terminales, formando un condensador. Entre medias y sin hacer contacto con ninguno de ellos se introduce el tercer cilindro, que está soldado al platillo y el conjunto platillo-cilindro está conectado a masa y suspendido por un muelle. De esta forma, al colocar un peso en el platillo, el tercer cilindro se desplaza hacia abajo, variando la capacidad del condensador formado por los otros dos. Al Aish y sus colegas utilizaron un microcontrolador PIC16F877 para procesar la componente en DC de la señal en cuadratura y así calcular la capacitancia del condensador. Sin embargo, en este caso la forma de obtener la componente en cuadratura es externa al microcontrolador, y se consigue de acuerdo a la figura 2.12.

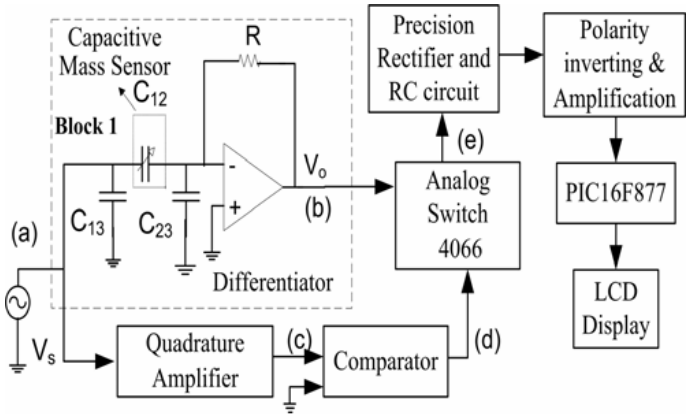


Figura 2.12: Esquema del sensor capacitivo de masa propuesto en [44].

2.3.2. Detectores de gas

Como complemento al artículo de Kolle y O'Leary sobre amplicación a sensores capacitivos, en [56], De Marcellis et al. proponen la aplicación de un amplificador Lock-in en el campo de los detectores de gas de tipo resistivo, llegando a aumentar la sensibilidad del sistema unas 80 veces.

[15]El PMMA, o Polimetil metacrilato, es el polímero más transparente que se conoce.

2.3.3. Espectroscopía de fluorescencia

Muchos instrumentos de laboratorio de uso común en química contienen un procesador digital de señales que incorporan un amplificador Lock-in. Un ejemplo son los espectrofotómetros de fluorescencia, en los que un haz de luz, comúnmente ultravioleta, irradia una muestra de una sustancia, generalmente disuelta. Las moléculas en disolución absorben entonces la energía de los fotones que inciden en ellas, pasando a un estado electrónico excitado. A continuación las moléculas decaen del estado excitado a alguno de los niveles vibratorios del estado fundamental emitiendo un fotón con una energía equivalente a la del salto; como existen varios niveles vibratorios en el estado fundamental, se emitirán fotones de diferentes longitudes de onda.

El haz de fotones emitidos por la muestra pasa por un monocromador, que internamente contiene una rejilla de difracción que lo descompone en varios haces de diferentes frecuencias, a modo de prisma. Una ranura móvil permite seleccionar únicamente uno de estos componentes. De esta forma, manteniendo la frecuencia de excitación de la muestra constante, es posible obtener un barrido de las frecuencias de emisión. Existen varios artículos recientes que discuten las ventajas de aplicar las técnicas de Lock-in a este tipo de espectroscopía.

2.3.4. Resonancia magnética nuclear

Los amplificadores Lock-in también tienen aplicación en el campo de la resonancia magnética nuclear. Estos instrumentos disponen de un generador de frecuencias que permite realizar cambios de fase sobre la señal. En un esquema típico, esta entra hacia la sonda donde se coloca la muestra. De esta forma, es posible medir, utilizando la frecuencia de referencia y la de salida de la sonda, las variaciones en la fase y la amplitud. Un artículo en el que se menta explícitamente una aplicación del Lock-in a RMN es el de Kan [70]

Figura 2.13: Montaje de un espectrómetro de resonancia magnetica nuclear (NMR) en el Laboratorio de Experimentación Avanzada Donald A. Glaser de la Universidad de California, Berkeley, Estados Unidos. Fotografía de Donald J. Tomado de [30].

2.3.5. Microscopía de fuerza atómica

La microscopía de fuerza atómica (AFM, *Atomic Force Microscopy* es una técnica inventada en 1986 por Gerd Binnig en 1986, mientras trabajaba para IBM. Poco después, Binnig y sus colegas Calvin Quate y Christoph Gerber implementaron físicamente el aparato [47]. Este dispositivo permite obtener imágenes con una resolución

El funcionamiento de un AFM es sencillo y está esquematizado en la figura 2.14. Se basa en una lámina de metal flexible, en uno de cuyos extremos hay una punta extremadamente afilada. Esta lámina, también llamada micropalanca, se posiciona con la punta dirigida hacia abajo y se conecta, por el extremo opuesto al de la punta, a un elemento piezoelétrico que puede hacer vibrar la lámina. La muestra se coloca por debajo de la punta, y un diodo láser emite un haz de luz que se refleja en el extremo detrás del cual está la punta y llega a una matriz de fotodiodos. Cuando la punta se mueve debido a la topología de la superficie de la muestra, el ángulo de reflexión cambia y el haz llega a un fotodiodo diferente, detectando el movimiento.

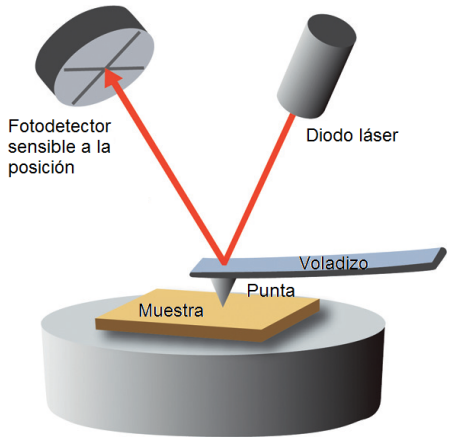


Figura 2.14: Funcionamiento de un microscopio de fuerza atómica. Tomado de [76].

Los microscopios de fuerza atómica son muy sensibles al ruido provocado por las vibraciones. Por este motivo estos instrumentos se montan sobre mesas estabilizadas de óptica. También debe prestarse especial cuidado al ruido provocado por variaciones en la temperatura, que pueden hacerlos vibrar a su frecuencia de resonancia.

Como hemos comentado anteriormente, en uno de los modos de trabajo, llamado 'modo dinámico' se hace vibrar la lámina. Dentro de éste, en el modo de frecuencia modulada, se hace vibrar la lámina a la frecuencia de resonancia de ésta. En este caso se pueden utilizar amplificadores Lock-in para mejorar las imágenes provenientes de este tipo de microscopio. Por un canal entra la señal de referencia, que es la de resonancia de la lámina, y por el canal de salida del sistema se conecta la salida de la matriz de fotodectectores, mejorando sensiblemente la calidad de la imagen. Dentro del 'modo dinámico' también existe el modo de 'amplitud modulada', en el que, de nuevo la lámina se hace vibrar, pero en esta ocasión manteniendo constante la amplitud en vez de la frecuencia.

Una variante de este sistema, llamada microscopía de fuerza atómica bimodal, hace resonar la

lámina utilizando dos de sus frecuencias de resonancia a la vez, una siendo la fundamental, y la otra un armónico de ésta [63]. Dentro del AFM bimodal, el modo de 'amplitud modulada' utiliza dos amplificadores Lock-in diferentes, uno para cada una de las dos frecuencias de resonancia, de las que se extraen el desfase y la amplitud de la señal de salida del sistema. De la componente a la frecuencia fundamental se extrae la topografía de la superficie del material que se está analizando, mientras que del armónico se extrae un mayor contraste.

2.3.6. Resonancia paramagnética electrónica

Mediante la Resonancia paramagnética electrónica, es posible detectar especies químicas con electrones desapareados, tales como radicales libres, o compuestos paramagnéticos, como son la mayoría de los que forman los metales de transición.

Figura 2.15: Espectrómetro EPR en la Universidad Jaguelónica, Polonia. Fotografía de Przemyslaw "Tukan" Grudnik, distribuida bajo licencia CC BY-SA. Tomado de [16].

Si observamos atentamente la figura 2.15, podemos ver que el segundo dispositivo empezando por arriba del armario azul de la izquierda es un Lock-in de SRS.

Este tipo de amplificadores también ha encontrado uso en el campo de la espectroscopía de resonancia magnética pulsada detectada eléctricamente (**pEDMR**, siglas de *pulsed Electrically Detected Magnetic Resonance*), que es un método indirecto de resonancia magnética electrónica mediante la medición de variaciones en la conductividad eléctrica. Este tipo de técnica tiene aplicación en el campo de los semiconductores, donde permite detectar la presencia de impurezas en muy bajas concentraciones, llegando incluso hasta algunos cientos de átomos [15]. Consiste en colocar la muestra que va a ser analizada en un resonador de microondas refrigerado por helio, que a su vez se sitúa en el campo magnético producido por dos bobinas de Helmontz, en una de las cuales hay una sonda de efecto Hall. A la muestra están conectados dos electrodos, que salen de la cámara de resonancia. Se hace pasar un voltaje a través de ellos y se generan pulsos de microondas en la cámara de resonancia, a una cierta frecuencia. A la vez se mide la corriente que circula a través de la muestra, utilizando un amplificador de transimpedancia. La frecuencia de referencia a la que se emiten los microondas y la señal de corriente procedente de la muestra pasan a un amplificador Lock-in, que envía los datos de

amplitud y desfase a un ordenador. A la vez, esa amplitud y desfase sirven para controlar la frecuencia del microondas, usando un modulador y un amplificador.

En el año 2012, Hoehne et al. publican un artículo [66] en el que demuestran que la aplicación de los Lock-in para pEDMR permite reducir el ruido de baja frecuencia un orden de magnitud.

2.3.7. Espectroscopía de impedancia electroquímica

Esta técnica está derivada de otra más general llamada 'espectroscopía de impedancia' o 'espectroscopía dieléctrica', y sirve para realizar las reacciones de corrosión sobre un material, generalmente un metal. Consiste en montar una celda electroquímica con tres electrodos, llamados electrodo de trabajo, de referencia, y auxiliar. Un aparato llamado potenciostato, aplica una señal en corriente alterna al electrodo de trabajo, fabricado del material bajo análisis, y mide su respuesta en intensidad.

Un segundo aparato, el Lock-in, se encarga de sintetizar la frecuencia y enviársela al potenciostato para que la aplique al electrodo, y también recibe la respuesta de éste, con la cual se calcula el desfase en la señal. A través de él obtiene la impedancia del sistema para esa frecuencia determinada. El proceso se repite para varias frecuencias diferentes, utilizando un ordenador para controlar el amplificador Lock-in, que en esta técnica se le llama 'Analizador de respuesta en frecuencia', abreviado FRA.

2.4. Solución matemática al problema planteado

Como hemos visto en la imagen 2.5 de la sección 2.2.3, un amplificador Lock-in multiplica la señal de referencia con la señal procedente de pasar la referencia por un proceso que lo desfasa. Para la multiplicación se utiliza un detector sensible a la fase (PSD), del cual se obtiene una señal con una componente de continua (que es proporcional al desfase) y una componente de alterna. La componente de alterna se descarta utilizando un filtro pasa baja. En este apartado se justifica, matemáticamente, este procedimiento.

El desarrollo de las ecuaciones que conforman las operaciones anteriores puede encontrarse en la literatura y realizarse considerando las ondas sinusoidales como senos [73] o como cosenos [52] [27], dando, en cualquiera de los casos, el mismo resultado, ya que un coseno es equivalente a un seno con un desfase de 90º, o expresándolo en radianes, $\frac{\pi}{2}$:

$$sin(x) = cos(x - \frac{\pi}{2}) \tag{2.16}$$

En nuestro caso, emplearemos el seno por sencillez, ya que resulta más sencillo que en un instante considerado como $t = 0$, el desfase del seno sea 0, y por tanto, los desfases de una onda respecto a otra se visualizan más fácilmente. La ecuación que define una onda sinusoidal es, entonces

$$x(t) = A \cdot sin(2\pi ft + \theta) = A \cdot sin(\omega t + \theta) \tag{2.17}$$

donde $x(t)$ es el valor del voltaje para un instante dado t, A es la amplitud de la onda, f es la frecuencia, ω es la velocidad angular, y θ es el desfase.

2.4.1. Esquema con un único PSD

Tal y como Leis et al. describen en [73], el funcionamiento de los amplificadores Lock-in puede explicarse con dos modelos; uno más sencillo que es el que aparece en la figura 2.16 y otro más complejo, que utiliza dos detectores sensibles a la fase y dos referencias (una tal cual y otra que igual a la anterior pero desfasada 90º), como se puede ver en las figuras 2.5 y 2.17. Por tanto, el SR850 utiliza el segundo modelo y será del que partamos para su implementación.

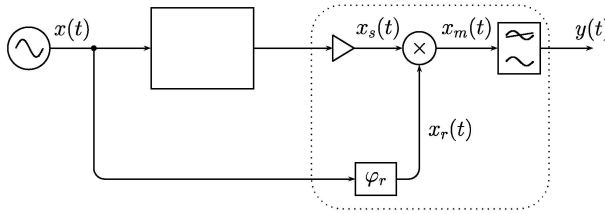


Figura 2.16: Configuración Lock-in de una sola referencia. Es necesario ajustar la fase de forma que la salida $y(t)$ sea máxima. Tomado de [73].

En nuestro sistema tenemos entonces dos señales sinusoidales, la de referencia, y la proveniente del dispositivo o sistema. Utilizaremos los subíndices r y s para referirnos a ellas, respectivamente. De esta forma, tenemos

$$x_r(t) = A_r \cdot sin\,(\omega_r t + \theta_r) \tag{2.18}$$

$$x_s(t) = A_s \cdot sin\,(\omega_s t + \theta_s) \tag{2.19}$$

Estas dos señales pasan al detector sensible a la fase, que las multiplica, efectuando la operación

$$x_m(t) = A_r \cdot sin\,(\omega_r t + \theta_r) \cdot A_s \cdot sin\,(\omega_s t + \theta_s) = A_r A_s sin\,(\omega_r t + \theta_r)\,sin\,(\omega_s t + \theta_s) \tag{2.20}$$

Para resolverla existe una igualdad trigonométrica que transforma un producto de senos en una suma:

$$sin\,(x)\,sin\,(y) = \frac{cos\,(x-y) - cos\,(x+y)}{2} \tag{2.21}$$

Si la aplicamos a la ecuación 2.20 (que se ha repetido en esta fórmula para mayor claridad) tomando $\omega_r t + \theta_r = x$ y $\omega_s t + \theta_s = y$ obtenemos que

$$x_m(t) = A_r A_s sin\,(\omega_r t + \theta_r)\,sin\,(\omega_s t + \theta_s) \tag{2.22}$$

$$= \frac{A_r A_s}{2}\,(cos\,((\omega_r t + \theta_r) - (\omega_s t + \theta_s)) - cos\,((\omega_r t + \theta_r) + (\omega_s t + \theta_s))) \tag{2.23}$$

Si ahora, como hemos comentado anteriormente, tenemos en cuenta que la señal de referencia se introduce en el dispositivo bajo pruebas y éste produce un desfase en ella pero no varía la frecuencia, o lo que es lo mismo $\omega_r = \omega_s$ (equivalentemente se dice que el sistema es lineal), la ecuación anterior queda de la siguiente forma:

$$x_m(t) = \frac{A_r A_s}{2}\,(cos\,(\omega_r t + \theta_r - \omega_r t - \theta_s) - cos\,(\omega_r t + \theta_r + \omega_r t + \theta_s)) \tag{2.24}$$

$$= \frac{A_r A_s}{2}\,(cos\,(\theta_r - \theta_s) - cos\,(2\omega_r t + \theta_r + \theta_s)) \tag{2.25}$$

Es decir, que obtenemos una componente de continua que es proporcional a la diferencia entre la fase de la referencia y la de la señal proveniente del dispositivo, y una señal en alterna proporcional al segundo armónico de la frecuencia de referencia, o sea, $2\omega_r t$. Utilizando un filtro pasa baja ideal, podemos eliminar la componente en alterna, quedándonos únicamente con la componente de continua:

$$y(t) = \frac{A_r A_s}{2} \cdot cos\,(\theta_r - \theta_s) \tag{2.26}$$

Si a continuación dispusiéramos de algún mecanismo que nos permitiera variar la fase proveniente de la señal de referencia, como por ejemplo mediante un desfasador sintonizable, se podría conseguir que $\theta_s = \theta_r$, momento que se detectaría cuando $y(t)$ tuviera sus valores máximos. Despejando obtendríamos:

$$y(t) = \frac{A_r A_s}{2} \cdot cos\,(0) \tag{2.27}$$

$$A_s = \frac{2y(t)}{A_r} \tag{2.28}$$

Esto es así ya que el conjunto imagen del coseno está en el rango $[0,1]$, de esta forma, el valor máximo se alcanza con el coseno de 0, que da un valor de 1, y por tanto, $y(t)$ sería máxima.

2.4.2. Esquema con dos PSDs

Sin embargo, el esquema de la sección anterior con un solo PSD añade el inconveniente de necesitar de un circuito adicional para variar la fase y detectar el valor máximo de $y(t)$, por ejemplo mediante un microprocesador y un conversor A/D. Para solucionar este problema existe el segundo modelo, que duplica el esquema anterior (utiliza dos PSDs y dos filtros pasa baja), alimentando ambos con la señal proveniente del sistema, a uno de ellos con la señal de referencia, y al otro, con esta misma señal de referencia desfasada 90º, como puede verse en la figura 2.17. De esta forma, a los

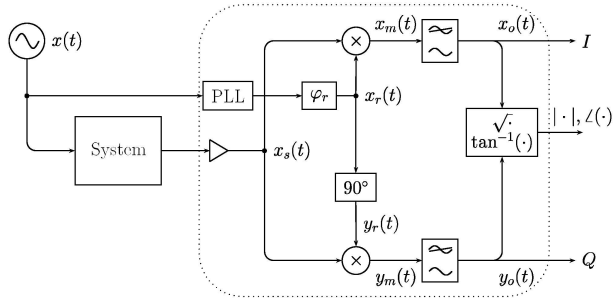


Figura 2.17: Configuración Lock-in de doble referencia. Tomado de [73].

PSDs entra, tanto la señal procedente del sistema $x_s(t)$, que es idéntica que en el modelo de Lock-in anterior, con la ecuación 2.19:

$$x_s(t) = A_s \cdot sin\,(\omega_s t + \theta_s) \tag{2.19 repetida}$$

como las señales $x_r(t)$, llamada señal en fase, y otra señal $x_r(t)$, que al haber sido desfasada 90º, se dice que está en cuadratura con la anterior. Teniendo en cuenta la equivalencia entre el seno y el

coseno expuesta en 2.16, las ecuaciones de ambas son, por tanto

$$x_r(t) = A_r \cdot sin(\omega_r t + \theta_r) \tag{2.29}$$

$$y_r(t) = A_r \cdot sin\left(\omega_r t + \theta_r + \frac{\pi}{2}\right) = A_r \cdot cos(\omega_r t + \theta_r) \tag{2.30}$$

Si ahora procedemos a realizar la multiplicación que tiene lugar en los PSDs, $x_m(t) = x_s(t) \cdot x_r(t)$ en el superior de la imagen 2.17, e $y_m(t) = x_s(t) \cdot y_r(t)$ en el inferior, obtenemos, respectivamente, las siguientes ecuaciones:

$$x_m(t) = A_r A_s \cdot sin(\omega_r t + \theta_r) sin(\omega_s t + \theta_s) \tag{2.31}$$

$$y_m(t) = A_r A_s \cdot cos(\omega_r t + \theta_r) sin(\omega_s t + \theta_s) \tag{2.32}$$

La primera ecuación se desarrolla y simplifica, igual que con el modelo de Lock-in anterior, utilizando la ecuación 2.33. Para la segunda necesitamos una equivalencia similar, que transforma la multiplicación de un coseno con un seno en una suma:

$$cos(x) sin(y) = \frac{sin(x+y) - sin(x-y)}{2} \tag{2.33}$$

PSD de la componente en cuadratura

Continuando con el desarrollo de la ecuación de la onda que sale del PSD inferior, $y_m(t)$:

$$y_m(t) = A_r A_s \cdot cos(\omega_r t + \theta_r) sin(\omega_s t + \theta_s) \tag{2.34}$$

$$= \frac{A_r A_s}{2} \cdot (sin((\omega_r t + \theta_r) + (\omega_s t + \theta_s)) - sin((\omega_r t + \theta_r) - (\omega_s t + \theta_s))) \tag{2.35}$$

$$\tag{2.36}$$

Volviendo a suponer que el dispositivo no modula la señal de referencia ($\omega_r = \omega_s$), o lo que es lo mismo, que la frecuencia no varía al pasar por él esa señal:

$$y_m(t) = \frac{A_r A_s}{2} \cdot (sin((\omega_r t + \theta_r) + (\omega_r t + \theta_s)) - sin((\omega_r t + \theta_r) - (\omega_r t + \theta_s))) \tag{2.37}$$

$$= \frac{A_r A_s}{2} \cdot (sin(2\omega_r t + \theta_r + \theta_s) - sin(\theta_r - \theta_s)) \tag{2.38}$$

Aplicando el filtro pasa baja para quedarnos con la señal en DC y descartar la componente de alterna (el segundo armónico de la frecuencia de referencia), obtenemos $y_0(t)$:

$$y_0(t) = \frac{A_r A_s}{2} \cdot sin(\theta_r - \theta_s) \tag{2.39}$$

PSD de la componente en fase

De la misma forma que con la componente en fase, y exactamente igual que en el desarrollo del esquema con un único PSD, podemos obtener la ecuación de la onda $x_m(t)$, a la salida del PSD superior:

$$x_m(t) = A_r A_s \cdot sin(\omega_r t + \theta_r) sin(\omega_s t + \theta_s) \tag{2.40}$$

$$= \frac{A_r A_s}{2} \cdot (cos((\omega_r t + \theta_r) - (\omega_s t + \theta_s)) - cos((\omega_r t + \theta_r) + (\omega_s t + \theta_s))) \tag{2.41}$$

Repitiendo la suposición de que el dispositivo bajo pruebas o el sensor no modifica la frecuencia de la señal de referencia ($\omega_r = \omega_s$) sabemos que:

$$x_m(t) = \frac{A_r A_s}{2} \cdot (cos((\omega_r t + \theta_r) - (\omega_r t + \theta_s)) - cos((\omega_r t + \theta_r) + (\omega_r t + \theta_s))) \quad (2.42)$$

$$= \frac{A_r A_s}{2} \cdot (cos(\theta_r - \theta_s) - cos(2\omega_r t + \theta_r + \theta_s)) \quad (2.43)$$

Y por último, aplicando el filtro pasa baja:

$$x_0(t) = \frac{A_r A_s}{2} \cdot cos(\theta_r - \theta_s) \quad (2.44)$$

Amplitud y desfase

Una vez obtenidas las componentes en fase y cuadratura, $x_0(t)$, e $y_0(t)$, es posible obtener la variación en la amplitud y en la fase que se ha producido en la señal de referencia cuando ha pasado a través del sistema que se está analizando. En los pasos anteriores hemos asumido que la frecuencia no se ve modificada, pero sí que pueden hacerlo la amplitud y la fase.

Con las ecuaciones anteriores es posible crear un sistema y resolver las incógnitas, que en este caso son A_s y θ_s, es decir, la amplitud y la fase que tienen la onda que sale del sistema. A_r y θ_r son la amplitud y fase de la señal de referencia, valores que conocemos dado que esa señal la generamos y controlamos nosotros. El sistema de ecuaciones a resolver es entonces:

$$y_0(t) = \frac{A_r A_s}{2} \cdot sin(\theta_r - \theta_s) \quad (2.39 \text{ repetida})$$

$$x_0(t) = \frac{A_r A_s}{2} \cdot cos(\theta_r - \theta_s) \quad (2.44 \text{ repetida})$$

Podemos empezar intentando separar de las ecuaciones los desfases despejando de ambas ecuaciones $(\theta_r - \theta_s)$, de la siguiente manera

$$sin^{-1}\left(\frac{2y_0(t)}{A_r A_s}\right) = \theta_r - \theta_s \quad (2.45)$$

$$cos^{-1}\left(\frac{2x_0(t)}{A_r A_s}\right) = \theta_r - \theta_s \quad (2.46)$$

Si ahora igualamos ambas ecuaciones obtenemos:

$$cos^{-1}\left(\frac{2x_0(t)}{A_r A_s}\right) = sin^{-1}\left(\frac{2y_0(t)}{A_r A_s}\right) \quad (2.47)$$

Esta ecuación puede seguirse desarrollando moviendo el arcoseno hacia la derecha:

$$sin\left(cos^{-1}\left(\frac{2x_0(t)}{A_r A_s}\right)\right) = \frac{2y_0(t)}{A_r A_s} \quad (2.48)$$

Existe una igualdad matemática sobre el seno de un arcocoseno, que se enuncia como

$$sin\left(cos^{-1}(x)\right) = \sqrt{1 - x^2} \quad (2.49)$$

Si aplicamos esa igualdad sobre la ecuación 2.48 y continuamos operando:

$$\sqrt{1-\left(\frac{2x_0(t)}{A_rA_s}\right)^2} = \frac{2y_0(t)}{A_rA_s} \tag{2.50}$$

$$1-\left(\frac{2x_0(t)}{A_rA_s}\right)^2 = \left(\frac{2y_0(t)}{A_rA_s}\right)^2 \tag{2.51}$$

$$1-\frac{(2x_0(t))^2}{(A_rA_s)^2} = \frac{(2y_0(t))^2}{(A_rA_s)^2} \tag{2.52}$$

$$(A_rA_s)^2-(2x_0(t))^2 = (2y_0(t))^2 \tag{2.53}$$

$$(A_rA_s)^2 = (2x_0(t))^2+(2y_0(t))^2 \tag{2.54}$$

$$(A_rA_s)^2 = 2^2x_0^2(t)+2^2y_0^2(t) \;=\; 2^2\left(x_0^2(t)+y_0^2(t)\right) \tag{2.55}$$

$$A_rA_s = \sqrt{2^2\left(x_0^2(t)+y_0^2(t)\right)} \;=\; 2\sqrt{x_0^2(t)+y_0^2(t)} \tag{2.56}$$

De donde finalmente, podemos obtener la amplitud de la onda que sale del sistema, A_s, con datos que ya conocemos, como son $x_0(t)$, $y_0(t)$ y A_r:

$$A_s = \frac{2}{A_r}\sqrt{x_0^2(t)+y_0^2(t)} \tag{2.57}$$

El procedimiento para calcular la fórmula del desfase de la señal que proviene del sistema, θ_s, es análogo al realizado para obtener el de la cuadratura, aunque un poco más sencillo. Se parte de nuevo de las ecuaciones de los desfases:

$$y_0(t) = \frac{A_rA_s}{2}\cdot sin(\theta_r-\theta_s) \tag{2.39 repetida}$$

$$x_0(t) = \frac{A_rA_s}{2}\cdot cos(\theta_r-\theta_s) \tag{2.44 repetida}$$

La diferencia con el caso anterior es que ahora, en vez de despejar los desfases, debemos despejar las amplitudes A_rA_s:

$$\frac{2y_0(t)}{A_rA_s} = sin(\theta_r-\theta_s) \tag{2.58}$$

$$\frac{2x_0(t)}{A_rA_s} = cos(\theta_r-\theta_s) \tag{2.59}$$

Lo que conseguimos desplazando al término derecho de la ecuación tanto $x_0(t)$ como $y_0(t)$:

$$\frac{2}{A_rA_s} = y_0(t)\cdot sin(\theta_r-\theta_s) \tag{2.60}$$

$$\frac{2}{A_rA_s} = x_0(t)\cdot cos(\theta_r-\theta_s) \tag{2.61}$$

Una vez hecho esto, podemos igualar ambas ecuaciones y continuar desarrollando para despejar

θ_s:

$$y_0(t) \cdot sin(\theta_r - \theta_s) = x_0(t) \cdot cos(\theta_r - \theta_s) \tag{2.62}$$

$$y_0(t) \cdot \frac{sin(\theta_r - \theta_s)}{cos(\theta_r - \theta_s)} = x_0(t) \tag{2.63}$$

$$y_0(t) \cdot tan(\theta_r - \theta_s) = x_0(t) \tag{2.64}$$

$$tan(\theta_r - \theta_s) = \frac{x_0(t)}{y_0(t)} \tag{2.65}$$

$$\theta_r - \theta_s = tan^{-1}\left(\frac{x_0(t)}{y_0(t)}\right) \tag{2.66}$$

$$-\theta_s = -\theta_r + tan^{-1}\left(\frac{x_0(t)}{y_0(t)}\right) \tag{2.67}$$

Finalmente, multiplicando ambos lados de la ecuación por -1, obtenemos la ecuación que nos da el desfase de la onda que sale del sistema, θ_s. Como en el caso anterior, depende únicamente de valores que o bien hemos fijado nosotros en la onda de referencia, o bien podemos medir con un voltímetro:

$$\theta_s = \theta_r - tan^{-1}\left(\frac{x_0(t)}{y_0(t)}\right) \tag{2.68}$$

2.4.3. Cálculo del desfase entre señales

Además de la forma expuesta en la ecuación 2.57, existe otra manera de obtener el desfase de una señal partiendo de la ecuación de la onda sinusoidal.

$$x(t) = A \cdot sin(\omega t + \theta) \tag{2.17 repetida}$$

Esta ecuación generalmente se emplea para calcular el valor que tiene una onda en un momento t, y con un desfase θ dados. Sin embargo, es posible interpretar la ecuación al revés; conocido el valor de voltaje o intensidad que toma la onda en un instante t, (y conociendo la velocidad angular ω) podemos obtener θ. Claro que también es posible eliminar la velocidad angular de la ecuación cuando $t = 0$, o lo que es lo mismo, en el instante inicial:

$$sin^{-1}\left(\frac{x(0)}{A}\right) = \theta \tag{2.69}$$

Si disponemos también de la señal de referencia, podemos suponer que $t = 0$ cada vez que ésta pasa por cero en entre el semiciclo negativo y positivo de la onda. Esto plantea el problema de que deberemos medir tanto los picos como los valles de la señal de referencia para detectar esta condición.

Existe otra forma de plantear este problema, aunque un detector de picos sigue siendo necesario como veremos más adelante. La idea consiste en que, si disponemos de los valores de la onda de referencia y de la que sale del sistema en un momento cualquiera, tomando que la onda de referencia tenga un desfase real de cero, podemos calcular el desfase que tiene en ese punto dado tanto la onda de referencia (llamémosle θ_i) como la onda de salida del sistema θ_j. Si θ_i no es cero, es porque en el instante en el que hemos medido los valores de ambas ondas, la señal de la referencia no estaba al

principio de cada periodo. Por tanto, para calcular el desfase real de la onda de salida del sistema, θ_s, basta con restar, del leído de esta onda, el extraído de la de referencia:

$$\theta_s = \theta_j - \theta_i \tag{2.70}$$

Este algoritmo es, de hecho, el que finalmente está implementada en el código debido a la forma en la que actúa el buffer del DAC del osciloscopio con el que debe funcionar la aplicación (Keysight DSO-X 3104A). A modo de ejemplo tanto del algoritmo como de un problema que se detectó en implementación, en la figura 2.18 se muestra una captura de pantalla de este instrumento. La señal proveniente del canal 1, en amarillo, es la señal de referencia que se genera desde el propio generador de ondas del osciloscopio. Esta señal pasa por el dispositivo bajo test, que la desfasa, y a continuación pasa por un amplificador SR445A. La señal de salida del amplificador entra al osciloscopio de nuevo por el canal 2 que se representa en verde.

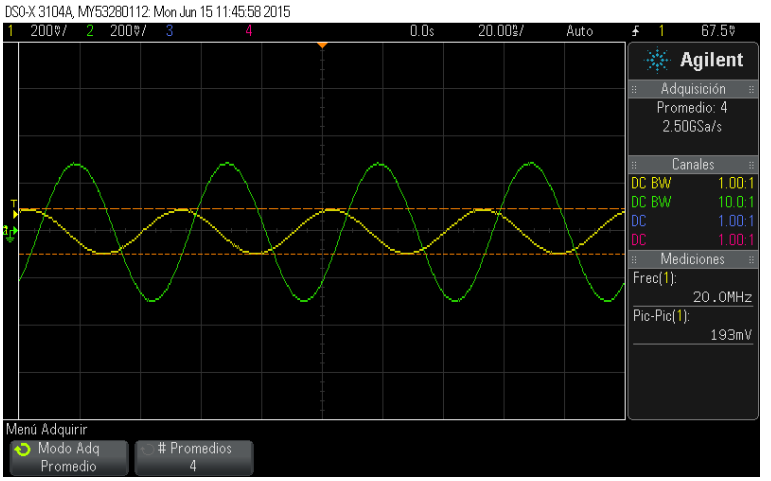


Figura 2.18: Captura del osciloscopio DSO-X 3104A, mostrando la señal de referencia en el canal 1 (amarillo) y la de salida del sistema, tras ser amplificada, en el canal 2 (verde). Se aprecia que, debido al corte hecho por el osciloscopio, la señal de referncia no empieza teniendo un valor 0.

La peculiaridad de este osciloscopio reside en que, aunque se envíe al DAC del osciloscopio una onda sinusoidal pura que haya sido previamente sintetizada con desfase cero, la función Auto Scale del osciloscopio puede empezar a mostrar (y por tanto cortar) la onda en cualquier punto.

Esto se traduce en que, si esa misma onda que se envía al generador de funciones se conecta directamente a uno de los canales sin pasar por un dispositivo que la modifique, y leemos los datos del búfer del ADC de ese canal, notaremos que el valor inicial de la onda de referencia no es cero aunque en los datos que se le envían al DAC sí lo sea, como puede verse en el extremo izquierdo de la imagen 2.18, donde la onda amarilla no empieza con un valor de 0. Esto provoca que la onda digitalizada (y visualizada) contenga un desfase ficticio θ_i respecto de la onda enviada, que se debe a este 'ventaneo' del osciloscopio.

Evidentemente, en cualquier captura que se realice, este desfase va a afectar por igual a las formas de onda capturadas en todos los canales, lo que hace necesario que al menos uno de ellos deba estar

conectado, directamente, a la salida del generador de funciones, con un desfase 0, para su uso como referencia. En este trabajo se adoptará la convención de que, cuando aparezcan dos o más canales en una imagen, el canal 1 siempre será la señal que sale del generador de funciones.

De esta forma, obteniendo el desfase que aparece en el canal 1 y sabiendo que la onda enviada no tenía desfase, disponemos θ_i y podremos restárselo a los desfases de todos los demás canales para obtener sus desfases reales, como hemos visto en la ecuación 2.70.

Sin embargo, el problema de la detección de picos vuelve a aparecer, y se debe a la forma en la que trabajan las funciones arcoseno, arcocoseno y arcotangente, cuya imagen está en el rango $\left[\frac{\pi}{2}, \frac{\pi}{2}\right]$. Si nos paramos a pensarlo detenidamente, la función seno obtiene la proyección en el eje y de un ángulo. Pero existe un problema: existen dos ángulos diferentes para los que la proyección en el eje y es la misma, que además son la 'imagen especular' con respecto a este eje. Podemos verlo mejor en la figura 2.19, donde están representados los ángulos 220° y 320°.

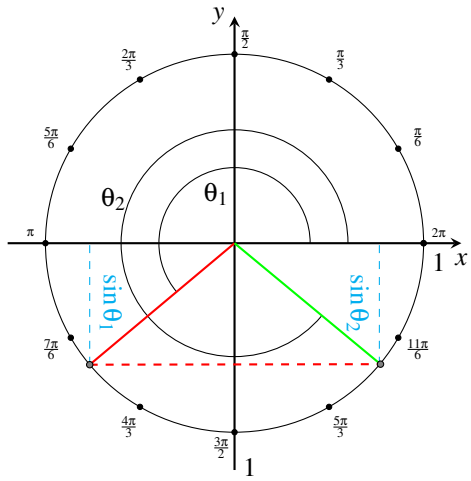


Figura 2.19: Círculo unidad, con los ángulos 220°, en rojo, y 320°, en verde, y su proyección en el eje y.

En efecto, la proyección de estos dos ángulos sobre este eje es la misma. Si hacemos la operación con una calculadora podemos verificar que son iguales:

$$sin(320^\circ) = sin(220^\circ) = -0{,}64278760968 \tag{2.71}$$

Por tanto, la función arcoseno debería devolver dos soluciones. Tradicionalmente devuelve las de la mitad derecha del círculo, esto es, entre $\left[\frac{\pi}{2}, \frac{\pi}{2}\right]$ pasando por 0, lo que matemáticamente se denomina rama principal del arcoseno. La solución para detectar cuál de los dos vectores es el correcto consiste en utilizar un detector de picos: buscaremos, en el vector de voltajes, la posición del siguiente pico y del siguiente valle. Si el próximo valle está más cercano al punto donde estamos midiendo el desfase, es que estamos descendiendo, y por tanto es necesario obtener el vector 'especular' del que nos devuelve la función. Veremos más acerca de esto en el capítulo 6.

2.4.4. Capacitancia y conductancia

Un desarrollo diferente para la obtención de la capacitancia y conductancia del circuito equivalente puede encontrarse en los artículos presentados por Abu Al Aish y sus colegas, en 2010 [44], y en el publicado por Kolle y O'Leary en 1998 [71].

Empecemos imaginando una onda sinusoidal pura, con cierto período T, representada en rojo en el siguiente diagrama. Se hace pasar esa onda por un circuito que provoca un desfase ϕ, dando como resultado la onda representada en azul:

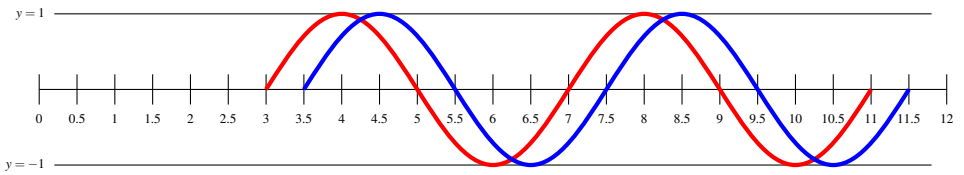


Figura 2.20: Circuito

Si empezamos a contar el tiempo desde 0 a partir de donde está marcado el eje horizontal, y fijamos t_0 para la onda de salida del sistema, tenemos que el desfase de ésta con respecto a la de referencia (en rojo) es igual a la diferencia entre el origen de coordenadas y el punto t_0, o lo que es lo mismo, $\theta = t_0$. De la misma forma, si establecemos que un periodo T equivale a 2π, es decir, $T = 2\pi$, entonces, podemos deducir que

$$\frac{t_0}{T} = \frac{\phi}{2\pi} \tag{2.72}$$

Si utilizamos las fórmulas de la componente en fase y cuadratura utilizando la intensidad de las señales, tenemos que:

$$I_{ph} = |I_m| \cdot cos(\phi) \tag{2.73}$$

$$I_q = |I_m| \cdot sin(\phi) \tag{2.74}$$

donde I_m es el valor de pico de la intensidad, es decir, la intensidad máxima. I_m es la amplitud de la onda sinusoidal.

Podemos obtener la intensidad que circula entre los extremos del circuito anterior multiplicando la diferencia de potencial entre los extremos del circuito por la impedancia equivalente:

$$I = V_m(G + j\omega C) \tag{2.75}$$

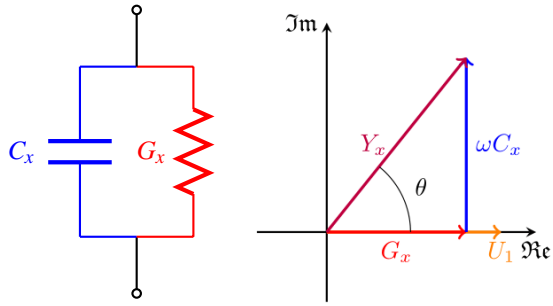
$$I_{ph} = V_m G \tag{2.76}$$

$$I_q = V_m \omega C \tag{2.77}$$

Si a continuación despejamos la conductancia (G) y la capacitancia (C) de las ecuaciones anteriores, obtendremos

$$G = \frac{I_{ph}}{V_m} = \frac{I_m \cdot cos(\theta)}{V_m} \tag{2.78}$$

$$C = \frac{I_q}{\omega V_m} = \frac{I_m \cdot sin(\theta)}{\omega V_m} \tag{2.79}$$



(a) Circuito equivalente (b) Diagrama de fasores

Figura 2.21: Circuito equivalente del D.U.T. y diagrama de fasores de éste. En azul se muestra la componente de capacitancia y en rojo, la componente de conductancia.

Capítulo 3
Gestión del proyecto

"La realidad no existe si no hay imaginación para verla."

Paul Auster

El modelo de análisis es la primera representación técnica de un sistema, y debe cumplir tres objetivos primarios y bien definidos:

- Describir lo que requiere el cliente.
- Establcer una base para la creación de un diseño de software.
- Definir un conjunto de requisitos que puedan validarse una vez construido el software.

Este análisis se ha realizado en base a observaciones en el uso de otras herramientas de gestión como Sigma, en una sesión de corrección con un OMR comercial, y en . Es importante no perder de vista que se trata de un sistema embebido, pero microcontrolado, que tiene mayor número de limitaciones que un ordenador de propósito general, sobre todo en aspectos de memoria y velocidad del procesador, y esto debe tenerse en cuenta a la hora de decidir qué clase de software se ejecutará sobre él.

3.1. Metodología empleada

Dado que en este caso se nos presentaba un proyecto mixto entre informática, electrónica y telecomunicaciones, hemos optado por una metodología ortodoxa. En lugar de seguir la metodología que plantea el proceso unificado, que solía ser lo habitual en las asignaturas de la Ingeniería Técnica en Informática, o de aplicar metodologías ágiles como SCRUM, hemos decidido tomar los elementos que más convenían al desarrollo de cada una e implementar nuestra propia metodología.

Esto se ha debido a varios factores. En primer lugar, en lugar de tener una fase de aprendizaje incial, para luego pasar a la fase de análisis, como sugiere el proceso unificado, hemos ido adquiriendo conocimientos a lo largo del proyecto, siendo algo más propio de procesos ágiles tipo SCRUM. Sin embargo, siendo únicamente dos desarrolladores en el proyecto y además disponiendo de tiempo limitado durante el primer cuatrimestre en mi caso, y en el segundo en caso de mi compañero, resultaba difícil repartir los roles que marca SCRUM entre nosotros.

El trabajo se ha realizado sobre una mesa larga que hay en el laboratorio 1L011 del Departamento de electrónica, sobre la cual hay un ordenador que ha sido adquirido a propósito para el para el proyecto, pero que deja espacio para tener, además, dos portátiles. El osciloscopio se ha dejado o bien en la misma mesa o bien en una mesa contigua, que permitía que el cable USB que va del propio osciloscopio al ordenador de sobremesa pudiera llegar, aunque un poco justo. Sobre esa mesa contigua se ha dejado el preamplificador, y la protoboard sobre la que se han configurado los diversos circuitos para obtener desfases y hacer pruebas se ha dejado en la mesa de desarrollo. En este sentido el entorno era un poco parecido a la metodología impulsada por XP (del inglés *eXtreme Programming*, programación extrema), que establece que lo ideal es que la programación se realice por parejas, con un ordenador por cada mesa, y mientras uno de los dos escribe, el otro descansa revisando el código que su compañero escribe, y al rato se cambian de rol, pasando a descansar el que escribe.

A medida que se iba aprendiendo y encontrando documentación y bibliografía, sobre todo en revistas científicas, se iban implementando diferentes funcionalidades y simplificando los requisitos, de tal forma que con apenas un número mínimo de ellos ha sido posible diseñar una aplicación con una flexibilidad extraordinaria, aunque algunas partes no han podido terminar de ser implementadas ya que el desarrollo se ha realizado por ciclos, hasta un total de cuatro a fecha de escritura de estas líneas. De esta forma, se ha ido dando forma a la aplicación a trozos, cambiando los requisitos, redistribuyendo la funcionalidad en clases diferentes. Ese ha sido otro de los motivos de que el Proceso Unificado no pudiera aplicarse. Sí que se ha seguido en cierto modo, en cada ciclo, las fases de análisis, diseño, e implementación, para llevar un cierto orden. Como se ha comentado previamente, se ha necesitado de cinco iteraciones para llegar a lo que se presenta en esta memoria, lo que ha supuesto refactorizar el código varias veces.

Sin embargo, y como se verá más adelante, se han utilizado técnicas de diseño centrado en el usuario ya que nuestros tutores representaban a la vez clientes de la aplicación. De esta manera, se ha realizado el diseño de las pantallas tomando como paradigma el trabajo común en un laboratorio de caracterización microelectrónica. Se han utilizado *mockups* para previsualizar el aspecto que tendrían las pantallas y revisar su adecuación al entorno.

Otra de las dificultades que hemos encontrado en el proyecto ha sido la de tener que realizar algo de trabajo de investigación (inventando algoritmos), bien para solucionar problemas que no aparecen matemáticamente, bien por la a nuestro juicio escasísima documentación de las bibliotecas IVI que controlan los instrumentos físicos, bien para optimizar los algoritmos de forma que que el software fuera lo más rápidamente posible para permitir un procesado lo más "en tiempo real" posible (procesamiento *on line*). En ocasiones eso ha supuesto modificar los diagramas en medio de un ciclo para poder continuar el desarrollo.

En el cambio desde la primera iteración a la segunda, se detuvo el proceso de desarrollo, se creó otro proyecto en Visual Studio, se creó el esquema de clases, se refactorizó el código para adaptarlo al nuevo proyecto y se continuó desarrollando. En cambio, en el paso de la segunda a la tercera, este cambio se realizó gradualmente, dejando de desarrollar la segunda versión del proyecto y empezando poco a poco a trabajar en la tercera. Durante la tercera, y dado que la funcionalidad de la clase onda tuvo que ser modificada radicalmente, lo cual supondría haber modificado gran parte del código (ya que el proyecto está centrado en el trabajo con las ondas), se decidió crear un proyecto aparte, de tal forma que pudiera seguirse desarrollando con el esquema de clases antiguo mientras se realizaban pruebas de la idoneidad del nuevo.

Una de las muchas ventajas del laboratorio 1L011 ha sido disponer de dos pizarras, una de vinilo blanco, donde hemos esbozado los algoritmos, diagramas de clase y de secuencia para tenerlos presentes mientras codificábamos y para exponérselos a la otra persona, y la otra de corcho, donde hemos improvisado un tablero KanBan, como puede verse en la figura 3.1. Esta segunda la hemos utilizado desde el comienzo del proyecto, mientras que la primera la empezamos a utilizar a principios de la tercera iteración. El problema de utilizar una pizarra de corcho ha sido tener que sujetar cada nota adhesiva con chinchetas, porque se despegaban. El motivo de haber utilizado la corchera como tablero Kanban se debe a su ubicación, ya que estaba ubicada delante de nosotros y podía tanto actualizarse con sólo levantarse, estirarse un poco y pinchar la nota, como saber el estado del proyecto simplemente levantando la vista. En cambio la otra no estaba fijada a la pared y podía dibujarse en ella con más facilidad.

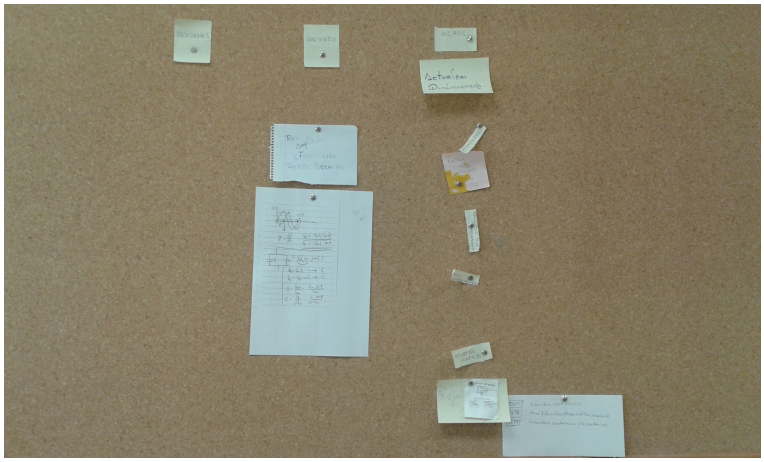


Figura 3.1: Estado del tablero KanBan improvisado en una corchera del laboratorio 1L011, durante la segunda iteración del proyecto.

Otra de las grandes ventajas de trabajar en este proyecto ha sido la posibilidad que nos han dado los tutores de utilizar el ordenador de sobremesa como un servidor permanente. Dados los problemas en los horarios que teníamos (en mi caso disponía de medio día para dedicarlo al proyecto durante el primer cuatrimestre), se nos ocurrió que sería buena idea instalar un sistema de control de versiones, como ha sido Team Foundation Server, que elegimos porque está pensado para funcionar con Visual Studio. De esta manera, aunque sólo hubiera una persona en el laboratorio, podía crear versiones nuevas y subir los cambios al servidor. En otras ocasiones, cuando la facultad estaba cerrada (por ejemplo, los fines de semana o vacaciones) nos era posible seguir trabajando cada uno desde su casa, utilizando el teléfono para coordinar qué parte desarrollaba cada cual. La posibilidad del servidor también nos ha permitido estar los dos conectados simultáneamente en el laboratorio realizando cambios en el proyecto; de no haber dispuesto de él solo podría estar escribiendo código a la vez.

Aun con todo lo expuesto anteriormente, esta memoria está presentada como si se hubiera seguido prácticamente un proceso unificado, con el objetivo de tener una mejor clasificación de la documentación. Lo que aquí se expone es una mezcla entre el análisis y el diseño resultantes de la cuarta iteración (punto en el que se ha detenido el desarrollo) y la implementación y pruebas de la tercera.

3.2. Participantes del proyecto

En este proyecto han participado, de forma directa, un total de cuatro personas, que en este ámbito se denominan skateholders. Helena Castán y Salvador Dueñas como tutores de proyecto, lo que en un proyecto de desarrollo podría considerarse como jefes de proyecto, marcando los tiempos, presentando y priorizando los objetivos y requisitos, realizando revisiones periódicas del estado del proyecto y explicando los fundamentos matemáticos. Por otra parte, también son los clientes de la aplicación, ya que el objetivo es que puedan utilizar el programa para caracterizar estructuras en materiales semiconductores, que es el ámbito de su grupo de investigación.

Por otra parte, mi compañero Andrés Escobar y yo hemos tomado el rol de desarrolladores de proyecto, escribiendo el código, buscando y leyendo bibliografía, realizando comentarios y sugerencias sobre los requisitos y la funcionalidad, instalando y configurando los equipos, etc.

Otras personas, como Jesús Arias y Héctor García han realizado aportaciones valiosas, sugiriendo y añadiendo bibliografía, o resolviéndonos dudas respecto del desarrollo o sobre fundamentos matemáticos.

3.3. Costes

En esta sección se comentan los costes que ha tenido el proyecto en términos de material hardware, herramientas software, salarios de las personas involucradas, y otros, como costes eléctricos. Algunos precios se han convertido de dólares USD a euros.

Se ha decidido incluir el coste de todos los materiales a pesar de que muchos de los aparatos físicos, como amplificadores Lock-in, generadores de ondas, el preamplificador y el osciloscopio ya formaban parte del grupo de investigación mucho antes de que se apareciera este proyecto. La decisión de incluirlos tiene como motivo ver cuáles habrían sido los costes reales de haberse llevado a cabo en una empresa que empezara su desarrollo y no dispusiera de ninguno de estos materiales.

Todos los precios utilizados en el cálculo de costes son reales y están extraídos de la página web del fabricante a fecha de redacción. En el caso de varios de ellos, como por ejemplo de Matlab, ya no están disponibles los precios de las versiones que se han utilizado, por lo que se han sustituido por los de la versión más próxima en el tiempo disponible (en el caso de Matlab, de la última a fecha de escritura de estas líneas, R2015a).

3.3.1. Material

En la sección de costes de material, distinguiremos entre costes de material hardware y software, ya que tienen características diferentes. Por ejemplo, lo normal es que un instrumento se pague una sola vez (a menos que se alguile), mientras que en software son más comunes las suscripciones.

Hardware

En este apartado se exponen los costes relacionados con el material físico que ha sido necesario para el proyecto, tanto para la simulación del amplificador Lock-in (lo que requiere de un ordenador), como para la conexión de los instrumentos al ordenador, y los instrumentos en sí. No se han tenido en cuenta algunos costes menores relacionados con las conexiones entre los dispositivos (cables BNC), el montaje del D.U.T. para realizar pruebas (como por ejemplo resistencias, condensadores y protoboards), etc.

Concepto	Proveedor	Precio
Ordenador Imaz i5-4460	Informática Imaz & Mate	694,23 €
Osciloscopio Keysight DSO-X 3104 A	Keysight Technologies	11642,00 €
Interfaz USB GPIB Keysight 82357B	Farnell	687,28 €
Preamplificador SRS SR445A	Stanford Research Systems	1182,10 €
Amplificador Lock-In SRS SR844	Stanford Research Systems	8324,00 €
Total		22529,61 €

Cuadro 3.1: Coste en productos hardware, en su mayoría instrumentación.

Software

Aquí se detallan los precios y cálculo del total relativo a los productos software utilizados.

En el caso de Visual Studio, gracias al programa DreamSpark al que está suscrito la escuela de informática, no ha sido necesario comprar la suite. En cuanto a Matlab, la UVa tiene licencia y es la que hemos utilizado durante el desarrollo, aunque como más adelante veremos, dejamos de utilizarle casi al principio debido a problemas en las bibliotecas. Team Foundation Server tiene la ventaja de ofrecerse de forma independiente a Visual Studio, aunque no entra dentro del programa DreamSpark.

Al igual que Matlab, la licencia de Office que se ha utilizado es la de la universidad; generalmente para tomar algunas notas para poderlas compilar en esta documentación del proyecto, aunque finalmente se ha empleado LaTeXpara redactarla.

La licendia de BenchLink Waveform Buider Pro, como en el caso de la instrumentación, ya había sido adquirida por el grupo de investigación antes de la aparición del proyecto.

3.3.2. Equipo humano

Como se ha expuesto en la sección 3.2, se cuenta con dos desarrolladores, que realizan parte de búsqueda de información, análisis y diseño, programación, y documentación de la aplicación, que para simplificar se contarán como analistas programadores. También forman parte del equipo dos jefes de proyecto.

Los precios se han extraído de documentos fechados entre 2012 y 2015, para que el cálculo sea lo más próximo a la realidad. Se tomará en consideración que el primer analista programador (Andrés) ha participado cuatro horas al día durante el segundo cuatrimestre, y el segundo (yo), ha participado

Concepto	Proveedor	Precio
Visual Studio Ultimate 2012	Microsoft	13299,00 €
Matlab & Simulink R2013b	MathWorks	69,00 €
Team Foundation Server 2012	Microsoft	93,26 €
Bonobo Git Server	Jakub Chodounsky	0 €
USB Monitor Pro	FabulaTech	152,60 €
Keysight IO Libraries Suite	Keysight Technologies	0 €
BenchLink Waveform Buider Pro	Keysight Technologies	690,00 €
Enterprise Architect 12.0	Sparx Systems	212,70 €
Balsamiq Mockups 3.1	Balsamiq Studios	79,70 €
Microsoft Office Professional Plus 2010	Microsoft	438,90 €
TexMaker 4.4.1	Pascal Brachet	0 €
MiKTeX 2.9.4248	Christian Schenk	0 €
Zotero 4.0.26.4	Roy Rosenzweig Center	0 €
Total		15035,16 €

Cuadro 3.2: Costes asociados a la adquisición de licencias para las herramientas software seleccionadas.

cuatro horas al día durante el primero. Los dos jefes de proyecto habrán estado a media jornada (cuatro horas al día). Los sábados y domingos se han tenido como no laborables y se han tenido en cuenta festivos. El desarrollo completo de la aplicación se toma como realizado entre el 22 de septiembre de 2014 (fecha de inicio del curso académico 2014/2015 en la Universidad de Valladolid) y el 21 de junio de 2015, aunque en realidad, se ha alargado un poco más.

La relación de días lectivos que se ha utilizado aparece en la tabla 3.3 y ha sido extraída del calendario académico oficial de la universidad. Febrero aparece dividido en dos por ser fin del primero cuatrimestre e inicio del segundo.

Mes	Días	Mes	Días
Setiembre	8	Febrero	(7+13)
Octubre	21	Marzo	20
Noviembre	20	Abril	17
Diciembre	13	Mayo	20
Enero	17	Junio	15

Cuadro 3.3: Relación de días lectivos según el calendario académico de la Universidad de Valladolid para el curso 2014/2015.

- ((86 días $\times$8 horas al día) + (85 días $\times$ 4 horas al día)) $\times$ 34,80 € por hora = 35.774,4 € programador 1.
- ((86 días $\times$4 horas al día) + (85 días $\times$ 8 horas al día)) $\times$ 34,80 € por hora = 35.635,2 € programador 2.
- 171 días $\times$4 horas al día $\times$ 50,00 € por hora = 34.200 € por cada jefe de proyecto software.

3.3.3. Otros costes

Dado que se ha empleado un ordenador como servidor, y por tanto encendido 24 horas al día, se añaden 83,42 € por costes eléctricos (84W de consumo del procesador, 6600 horas encendido, 0.124368 € por kWh).

3.3.4. Coste total

Sumando todos los apartados anteriores, el coste total de este proyecto ascendería a 143.257,79 €.

3.4. Herramientas utilizadas

A continuación procederemos a resumir brevemente algunas de las herramientas que se han empleado para la realización del proyecto

3.4.1. Visual Studio 2012

Visual Studio es un entorno de programación integrado (IDE, *Integrated Development Environment*, que permite desarrollar código en varios lenguajes (entre ellos, C++, C#, VB.NET, F# y JavaScript), compilarla y verificarla. También es muy eficaz a la hora de tener una buena clasificación de los ficheros de código fuente, ya que permite crear carpetas conceptuales, que luego transforma en carpetas del sistema de ficheros.

La forma de crear un formulario es simple: basta con arrastrar los controles (también llamados *widgets*), tales como *comboboxes*, *textboxes*, botones, etc, desde el menú 'Cuadro de herramientas', y ajustar algunos parámetros, relacionados tanto con la apariencia (p. ej. el color), como con el comportamiento (definiendo a qué función de nuestro código debe llamarse cuando se produzca un determinado evento, como por ejemplo una pulsación simple con el botón izquierdo del ratón).

Otra de sus grandes ventajas es que permite dos modos de compilación: 'Debug' y 'Release'. En el primero se incluyen dentro del ejecutable los símbolos de depuración, que permiten que un *debugger* pueda tener acceso a más información sobre la aplicación, lo que permite una mejor depuración en caso de producirse una excepción, mostrando por ejemplo la pila de llamadas. Normalmente cuando se compila una aplicación, los nombres de las funciones internas desaparecen, mientras que haciéndolo en modo 'Debug' se mantienen. Evidentemente, esto provoca que los ejecutables ocupen más espacio.

En este proyecto hemos utilizado Visual Studio 2012, y hemos podido probar la versión ultimate, que permite extraer del código fuente el diagrama de clases y el de dependencias.

El lenguaje elegido para este proyecto fue Visual C#, dado que mi compañero Andrés había desarrollado en él su proyecto de fin de carrera y en mi caso no me importaba adaptarme a un lenguaje nuevo, ya que en el transcurso de la carrera se aprende a utilizar C y en el máster resulta necesario el uso de Java para la optativa de WEB, y C# es similar a ambos.

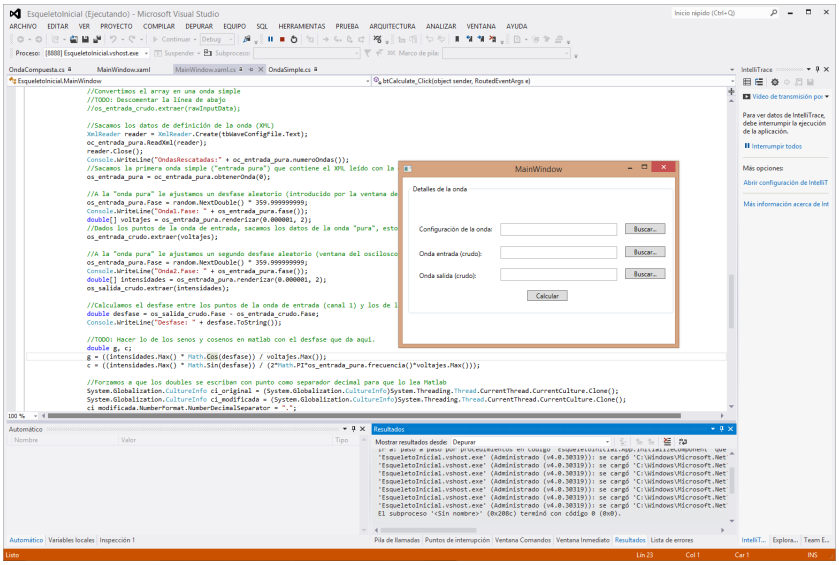


Figura 3.2: Interfaz del IDE Visual Studio 2012, versión Ultimate, depurando un programa.

3.4.2. Matlab

Matlab es un lenguaje inicialmente orientado a cálculos matemáticos (de hecho Matlab es la abreviatura de *MATrix LABoratory*, aunque actualmente dispone de muchas bibliotecas que permiten extender su funcionalidad, entre las cuales pueden citarse las de inteligencia artificial, biología computacional, procesado de datos, ...

La desventaja de este software es el elevado espacio que consume en el disco duro (9 GB en una instalación con todas las bibliotecas disponibles), y los recursos que necesita para ejecutarse.

Respecto a la interfaz, está diseñada de forma muy intuitiva. Por defecto, en la parte central de la pantalla está la ventana de comandos. Como éste es un lenguaje interpretado, se muestra el *prompt*, que es >>, al comienzo de cada línea tras pulsar intro. Si la llamada a una función o comando tiene salida, es posible suprimirla terminando la línea con punto y coma (';'). A la izquierda está situado un explorador de archivos, que permite ejecutar con doble *click* los scripts hechos para este lenguaje, cuya extensión de nombre de archivo es '.m'. A la derecha, arriba, disponemos del espacio de trabajo, en el que se pueden ver las variables que han sido creadas y sus tamaños (lo que es útil en caso de trabajar con matrices). Por último, debajo del espacio de trabajo tenemos el histórico de comandos, y podemos volver a lanzarlos con solo seleccionarlos y pulsar la tecla F9 del teclado.

En un principio se empezó utilizando como lenguaje de programación para el proyecto, pero se acabó por descartar debido a que éramos incapaces de controlar el generador de ondas con la interfaz provista por los controladores IVI, aparentemente por no disponer de esa funcionalidad implementada, que resulta clave para el proyecto, como veremos más adelante.

Sin embargo, ya que se disponía de él, se ha continuado utilizando para depurar en algunos momentos del proyecto, importando vectores de datos provenientes del ADC del osciloscopio, para ve-

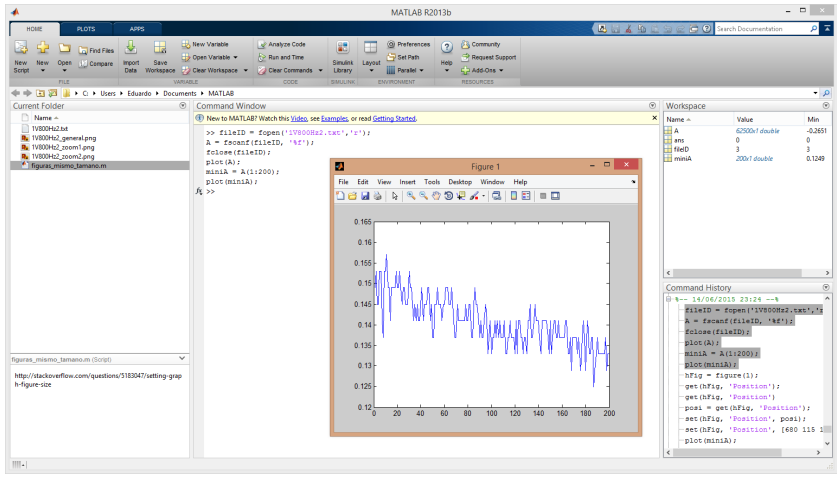


Figura 3.3: Interfaz del paquete de software Matlab, mostrando un gráfico.

rificar el buen funcionamiento del componente que permite a nuestro proyecto visualizar las ondas, y para visualizar los datos recibidos de los canales y enviados al ADC con una mejor resolución, ya que la función plot de Matlab ajusta automáticamente las escalas y también permite realizar ampliación y reducción de los gráficos. Algunas imágenes de esta memoria han sido obtenidas con Matlab.

3.4.3. Team Foundation Server

Dado que se ha dispuesto de un ordenador funcionando a modo de servidor en el laboratorio donde se ha realizado el proyecto, y que hemos sido dos desarrolladores subiendo y modificando código, resulta obligatorio disponer de algún sistema de control de versiones (CVS, Concurrent Versioning System). Aunque en los primeros momentos del proyecto estuvimos empleando el correo electrónico para mandar modificaciones, resulta incómodo ya que obliga a una numeración estricta, y a fusionar los ficheros y su contenido de forma manual.

Ya que Visual Studio 2012 tiene integrados el sistema Team Foundation Server por defecto, que es una solución propietaria de Microsoft, se optó por utilizar este segundo ya que, lógicamente, se espera un menor número de incompatibilidades y una mayor facilidad de uso al estar desarrollados ambos sistemas (Visual Studio y Team Foundation Server) por la misma compañía.

Team Foundation Server utiliza el servidor HTTP Internet Information Services (IIS), y por defecto se conecta con el servidor a través del puerto 8080.

Uno de los inconvenientes que encontramos es que el modo más común y documentado de gestionar usuarios asociados a TFS (esto es, aquellos que pueden subir, modificar y editar código) es asociándolos a las cuentas de usuario existentes en el equipo con Windows en el que está instalado el servidor. De esta forma, tuvimos que crear una cuenta para cada uno en el sistema.

3.4.4. Bonobo Git Server

Este software es, como su propio nombre indica, un servidor GIT para plataformas Windows con Internet Informetion Services (IIS). Fue creado en octubre de 2011 por Jakub Chodounsky, y a fecha de redacción, se encuentra en la versión 4.0.0. Está disponible bajo licencia de código abierto MIT.

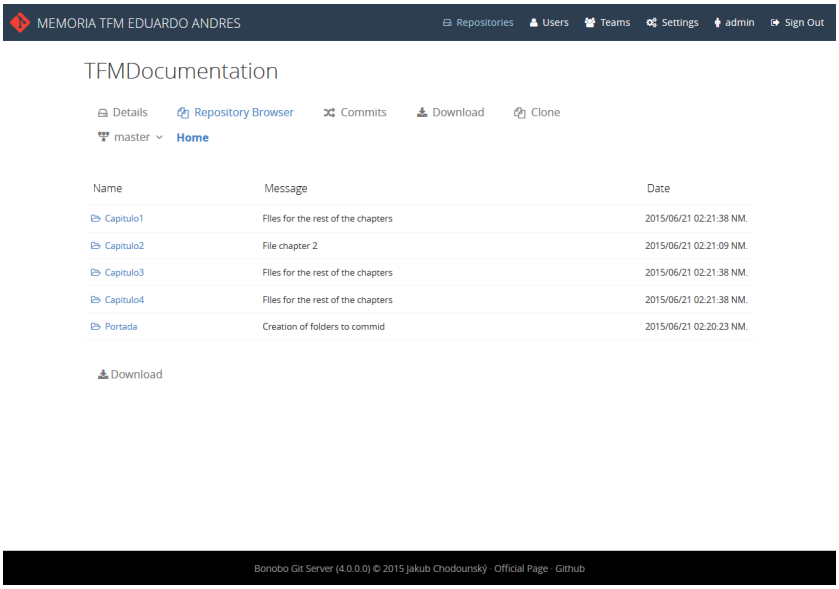


Figura 3.4: Interfaz web del servidor GIT Bonobo.

La razón de necesitar este software fue que en un momento del desarrollo, perdimos conectividad con el servidor Team Foundation Server (no pudiendo conectarnos a través del puerto 8080 de la máquina), tanto desde las redes de la universidad como desde casa, lo que nos obligó a buscar otro sistema que pudiera estar soportado por Visual Studio. A través de varias referencias encontradas en Internet, decidimos probar con el servidor Git Bonobo. Realizamos varios intentos para configurar nuestros portátiles con este software, pero resultaron infructuosos.

Finalmente, hablamos con el administrador de redes del departamento, que nos asignó una IP nueva y nos abrió el puerto 8080, de forma que volvimos a utilizar TFS como sistema de control de versiones. Posteriormente volvimos a utilizarlo como sistema de compartición de archivos que no fueran de código fuente; por ejemplo archivos de texto con referencias útiles, diagramas UML, etc.

Como punto fuerte, cabe resaltar que la instalación de Bonobo es muy sencilla; simplemente se descarga desde la página oficial un archivo comprimido 'Bonobo.Git.Server.4.0.0.zip', se descomprime, y la carpeta resultante, llamada Bonobo.Git.Server se mueve al directorio donde residen los proyectos HTTP de Microsoft IIS, que es C:\inetpub\wwwroot. Al marcar la casilla 'Modificar' se activa automáticamente la casilla 'Escritura'. Por último, basta con acceder al panel de control de IIS, y transformar la carpeta de Bonobo en una aplicación web. La desinstalación es también muy simple y consiste en realizar el proceso inverso.

3.4.5. USB Monitor Pro

A la mitad del proyecto nos dimos cuenta de que no éramos capaces de que una onda sinusoidal simple enviada por nuestro software (y por tanto con el modo del generador de funciones seleccionado en 'Arbitraria') encajara perfectamente con una generada por el propio osciloscopio (seleccionado en ?Seno?). Al probar con el software Benchlink Waveform Builder, para el que disponíamos de licencia, y ver que éste sí era capaz de generar ondas idénticas a las generadas en modo 'Seno' por el aparato, decidimos que resultaría conveniente saber qué comandos y cómo le estaba enviando los datos al osciloscopio la aplicación Benchlink.

Ya que la conexión con el osciloscopio Keysight DSO-X 3104 A es a través de USB, utilizamos el software USB Monitor Pro, que sirve para capturar (*sniffing*) el tráfico entre el anfitrión y los dispositivos conectados a este bus.

Capítulo 4
Análisis

"Anticiparse a los problemas y averiguar cómo resolverlos es, de hecho, lo contrario a alarmarse: es productivo."

Chris Hadfield

El modelo de análisis es la primera representación técnica de un sistema, y debe cumplir tres objetivos primarios y bien definidos:

- Describir lo que requiere el cliente.
- Establcer una base para la creación de un diseño de software.
- Definir un conjunto de requisitos que puedan validarse una vez construido el software.

Este análisis se ha hecho en base a lo que nos ha solicitado el cliente, en este caso nuestros tutores, y a lo que hemos podido apreciar en su trabajo diario en el laboratorio 1L012. Dado que la idea es que el sistema funcione sobre el equipo del que disponemos en el laboratorio contiguo (1L011), hemos agregado algunas de las características de éste como mínimos a cumplir, sin que esto suponga prejuicio para que también pueda ser extendido a otras plataformas, o al menos ser diseñado para ello.

4.1. Objetivos

Al igual que se comentó en el primer capítulo de la memoria, el objetivo de este proyecto es crear un sistema que permita emular completamente un Amplificador Lock-in. Para poderle simular plenamente, también resulta necesario disponer de varias fuentes de datos de entrada, lo que se logra bien conectando el Lock-in virtual a un generador de funciones, ya sea virtual o físico, o a un osciloscopio.

El aspecto interesante de este proyecto es poder controlar la señal que atraviesa el dispositivo bajo test, motivo por el cual el sistema a crear debe ser capaz de crear y configurar una señal de entrada, renderizarla a través de un generador de señales (físico o virtual), simular u obtener un desfase producido por un dispositivo bajo test, recoger esta señal y pasarla como segundo canal al amplificador Lock-in.

Los objetivos detectados se resumen, entonces, en los siguientes:

OBJ-001	Conexión y trabajo con instrumentos reales
Descripción	El sistema deberá ser capaz de conectarse físicamente con diversa instrumentación de laboratorio de electrónica, reconocerlo, configurarlo y mandarle órdenes. También debe ser capaz de recibir y decodificar correctamente la información enviada desde los éstos al sistema.
Estabilidad	Alta

OBJ-002	Emulación de instrumentos
Descripción	El sistema deberá poder emular, mediante software, el comportamiento de diversos instrumentos de laboratorio de electrónica, como amplificadores Lock-in, osciloscopios y generadores de ondas. También debe permitir la conexión entre éstos.
Estabilidad	Alta

Tabla 4.2: OBJ-002: Emulación de instrumentos

OBJ-003	Configuración y síntesis de formas de onda
Descripción	El sistema deberá permitir la definición de múltiples formas de onda y su discretización. De la misma manera, el sistema debe ser capaz de extraer las definiciones de las formas de onda a partir de una serie de datos discretos.
Estabilidad	Alta

Tabla 4.3: OBJ-003: Configuración y síntesis de formas de onda

OBJ-004	Metodología de trabajo
Descripción	El sistema deberá presentar una metodología de trabajo lo más próxima a la que realiza un investigador en el laboratorio.
Estabilidad	Alta

Tabla 4.4: OBJ-004: Metodología de trabajo

4.2. Especificación de requisitos software

La especificación de requisitos software es una descripción de las características que debe o se aconseja que tenga sistema que se va a desarrollar. Esta descripción debe ser completa y expresarse con exactitud para evitar el riesgo de que sea interpretada de forma diferente por usuarios y desarrolladores. Se deben incluir tanto los requerimientos del usuario para el sistema como una especificación detallada de los requerimientos del sistema.

4.2.1. Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones concretas.

FRQ-001	**Conexión y control de generador de ondas**
Objetivos de los que depende	**OBJ-001** Conexión y trabajo con instrumentos reales **OBJ-003** Configuración y síntesis de formas de onda
Descripción	El sistema deberá ser capaz de conectarse a un dispositivo que permita generar frecuencias y enviarle la configuración de una forma de onda definida por el usuario, para que el dispositivo convierta la configuración en voltaje.
Comentarios	Ninguno

Tabla 4.5: FRQ-001 Conexión y control de generador de ondas

FRQ-002	**Conexión a osciloscopio**
Objetivos de los que depende	**OBJ-001** Conexión y trabajo con instrumentos reales **OBJ-003** Configuración y síntesis de formas de onda
Descripción	El sistema deberá ser capaz de conectarse a un osciloscopio o equivalente que disponga de un DAC, y obtener de él los valores de voltaje en digital, de al menos dos canales.
Comentarios	Ninguno

Tabla 4.6: FRQ-002 Conexión a osciloscopio

FRQ-003	**Generación de formas de onda simples y compuestas**
Objetivos de los que depende	**OBJ-003** Configuración y síntesis de formas de onda
Descripción	El sistema deberá ser capaz de generar formas de onda, tanto simples (formadas por una única onda sinusoidal) como compuestas, permitiendo elegir la amplitud y la frecuencia de sus componentes.
Comentarios	Ninguno

Tabla 4.7: FRQ-003 Generación de formas de onda simples y compuestas

FRQ-004	**Tipos de análisis**
Objetivos de los que depende	**OBJ-003** Configuración y síntesis de formas de onda

Descripción	El sistema será capaz de realizar tres tipos de análisis: sinusoidal puro, barrido multifrecuencial y con forma de onda compleja.
Comentarios	Ninguno

Tabla 4.8: FRQ-004 Tipos de análisis

FRQ-005	**Obtención de la componente en fase**
Objetivos de los que depende	**OBJ-002** Emulación de instrumentos
Descripción	El sistema deberá ser capaz de obtener la componente en fase de la onda sinusoidal resultante de pasar la forma de onda generada por un dispositivo bajo pruebas (D.U.T., device under test), lo que genera un desfase sobre ésta última.
Comentarios	Ninguno

Tabla 4.9: FRQ-005 Obtención de la componente en fase

FRQ-006	**Obtención de la conductancia**
Objetivos de los que depende	**OBJ-002** Emulación de instrumentos
Descripción	El sistema deberá poder obtener, a partir de los datos de la componente en fase, la conductancia del D.U.T.
Comentarios	Ninguno

Tabla 4.10: FRQ-006 Obtención de la conductancia

FRQ-007	**Obtención de la componente en cuadratura**
Objetivos de los que depende	**OBJ-002** Emulación de instrumentos
Descripción	El sistema deberá ser capaz de obtener la componente en cuadratura de la onda sinusoidal resultante de pasar la forma de onda generada por un dispositivo bajo pruebas (D.U.T., device under test), lo que genera un desfase sobre ésta última.
Comentarios	Ninguno

Tabla 4.11: FRQ-007 Obtención de la componente en cuadratura

FRQ-008	Obtención de la capacitancia
Objetivos de los que depende	**OBJ-002** Emulación de instrumentos
Descripción	El sistema deberá poder obtener, a partir de los datos de la componente en cuadratura, la capacitancia del D.U.T.
Comentarios	Ninguno

Tabla 4.12: FRQ-008 Detección de marcas

4.2.2. Requisitos no funcionales

Los requisitos no funcionales de un sistema son aquellos requerimentos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste, como pueden ser la fiablidad o el tiempo de respuesta, entre otras.

NFR-001	Osciloscopio DSO-X 3104A como generador de señal
Objetivos de los que depende	**OBJ-001** Conexión y trabajo con instrumentos reales
Requisitos de los que depende	Ninguno
Descripción	El sistema deberá soportar, al menos, el osciloscopio DSO-X 3104A para generar las frecuencias.
Comentarios	Ninguno

Tabla 4.13: NFR-001 Osciloscopio DSO-X 3104A como generador de señal

NFR-002	Osciloscopio DSO-X 3104A como conversor ADC
Objetivos de los que depende	**OBJ-001** Conexión y trabajo con instrumentos reales
Requisitos de los que depende	Ninguno
Descripción	El sistema deberá soportar, al menos, el osciloscopio DSO-X 3104A para digitalizar los canales de entrada.
Comentarios	Ninguno

Tabla 4.14: NFR-002 Osciloscopio DSO-X 3104A como ADC

4.3. Modelo de casos de uso

Utilizamos los casos de uso para describir la forma en la que los usuarios interaccionan con nuestra aplicación. En los casos de uso se detallan paso a paso las acciones que realiza el sistema y que producen un resulado para un actor en particuar.

Los casos de uso y actores identificados durante esta fase se describen en el siguiente diagrama:

NFR-003	Keysight 82357B como interfaz USB 2.0 con Lock-In GPIB
Objetivos de los que depende	**OBJ-001** Conexión y trabajo con instrumentos reales
Requisitos de los que depende	Ninguno
Descripción	El sistema deberá soportar el uso de la interfaz hardware Keysight 82357B entre el puerto USB 2.0 del ordenador y el GPIB del Lock-in a controlar.
Comentarios	Gran parte de los amplificadores e instrumentación electrónica en general utiliza el puerto GPIB, sobre todo aquellos con cierta antigüedad.

Tabla 4.15: NFR-003 Keysight 82357B como interfaz USB-GPIB

NFR-004	Soporte del Amplificador Lock-In SR844
Objetivos de los que depende	**OBJ-001** Conexión y trabajo con instrumentos reales
Requisitos de los que depende	Ninguno
Descripción	El sistema deberá poder controlar, al menos, el amplificador Lock-in modelo SR844 de Stanford Research Systems.
Comentarios	Ninguno

Tabla 4.16: NFR-004 Soporte del Lock-In SR844

4.3.1. Actores del sistema

Los actores son todas aquellas entidades externas que puedan interactuar con la aplicación. Son, por tanto, los distintos papeles o roles que pueden tener los usuarios. Se debe tener en cuenta que el propio sistema puede ser un actor.

NFR-005	Lenguaje de programación Matlab, C o derivados
Objetivos de los que depende	Ninguno
Requisitos de los que depende	Ninguno
Descripción	El sistema deberá ser implementado utilizando el lenguaje Matlab, C, o alguno de sus derivados, como C++ y C#.
Comentarios	Se prefiere C o alguno de sus derivados para posibilitar que el sistema pueda ser portado a C o C++ y compilado para un ADC y para diversos sistemas operativos.

Tabla 4.17: NFR-005 Lenguaje de programación

NFR-006	Sistema operativo Windows 8.1
Objetivos de los que depende	Ninguno
Requisitos de los que depende	Ninguno
Descripción	El sistema deberá poder ejecutarse, al menos, sobre el sistema operativo Microsoft Windows 8.1
Comentarios	Debido a que ordenador disponible para desarrollo en el laboratorio 1L011 tiene esta característica

Tabla 4.18: NFR-006 Sistema operativo Windows 8.1

NFR-006	XML como estructura de almacenamiento
Objetivos de los que depende	Ninguno
Requisitos de los que depende	Ninguno
Descripción	Las formas de ondas generadas deberán poder ser almacenadas en y leídas desde ficheros estructurados como XML.
Comentarios	De esta forma es posible trabajar con ellas (crearlas y modificarlas) desde otros programas, incluso desde un editor de textos.

Tabla 4.19: NFR-007 XML como estructura de almacenamiento

ACT-001	Usuario
Descripción	Este actor representa de forma genérica a cualquier actor que utilice el sistema
Comentarios	Ninguno

Tabla 4.20: ACT-001 Usuario

ACT-002	Instrumento físico
Descripción	Este actor representa a un escáner que se conecte al sistema, y al que el sistema solicitará las imágenes.
Comentarios	Este actor es secundario en el sistema

Tabla 4.21: ACT-004 Instrumento físico

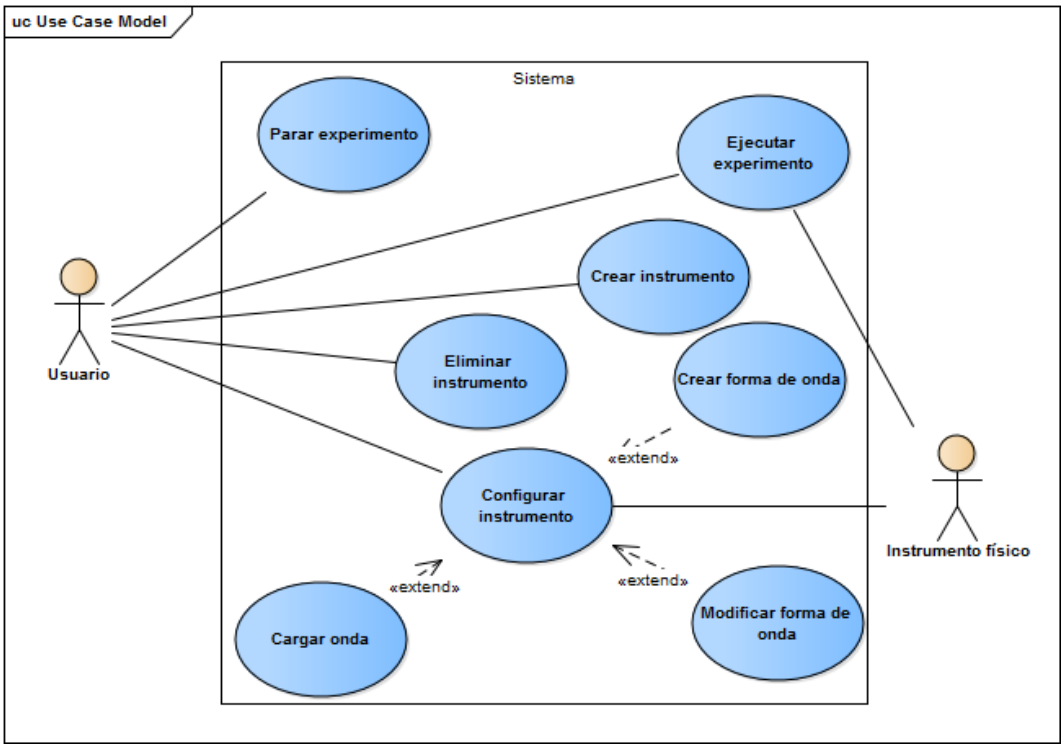


Figura 4.1: Diagrama de casos de uso

4.3.2. Especificación de los casos de uso

En este apartado se define la interacción entre los distintos acciones con el sistema para llevar a cabo el objetivo definido por el caso de uso, como una secuencia. Cada caso de uso lleva un identificador único, y depende de una serie de objetivos definidos anteriormente.

CU-001 Crear un instrumento

CU-001	**Crear un instrumento**	
Dependencias	**OBJ-002** Emulación de instrumentos	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera agregar una nueva unidad de un instrumento concreto al experimento	
Precondiciones	El sistema debe encontrarse iniciado	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) indica que se dispone a añadir al sistema un nuevo instrumento
	2	El sistema muestra todos los instrumentos disponibles al usuario, clasificados por tipo, marca y modelo
	3	El actor Usuario (ACT-001) selecciona el instrumento que desee añadir.

	4	El sistema verifica cuantos instrumentos de ese tipo están activos
	5	El sistema crea y registra una nueva unidad del instrumento que ha seleccionado
	6	El sistema asigna a la nueva unidad del instrumento un identificador único basado en su tipo
	7	El sistema marca todos los canales de entrada y salida de la nueva unidad del instrumento como desconectados
	8	El sistema muestra la unidad recién creada del instrumento al usuario
	9	El caso de uso finaliza
Postcondición	Se ha creado una unidad de un instrumento del tipo especificado por el usuario	
Comentarios	Ninguno	

Tabla 4.22: CU-001 Crear un instrumento

CU-002 Eliminar un instrumento

CU-002	**Eliminar un instrumento**	
Dependencias	**OBJ-001** Conexión y trabajo con instrumentos reales **OBJ-002** Emulación de instrumentos	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera eliminar una unidad concreta de un instrumento del experimento	
Precondiciones	El sistema debe encontrarse iniciado	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) solicita al sistema la lista de instrumentos creados
	2	El sistema muestra todos los instrumentos creados al actor Usuario (ACT-001)
	3	El actor Usuario (ACT-001) selecciona el instrumento que desee eliminar.
	4	El sistema muestra al actor Usuario (ACT-001) el instrumento seleccionado
	5	El actor Usuario (ACT-001) indica al sistema que desea eliminar el instrumento mostrado
	6	Si el instrumento dispone de canales de entrada, el sistema crea una lista E con los instrumentos a los que el que el que va a ser eliminado está conectado.
	7	Si el instrumento dispone de canales de salida, el sistema busca entre todos los instrumentos, aquellos que tengan canales de entrada conectados al instrumento que va a ser eliminado, y crea con ellos la lista S.
	8	El sistema muestra al actor Usuario (ACT-001) las listas E y S, y le consulta si realmente desea eliminar el instrumento.
	9	El actor Usuario (ACT-001) decide si desea eliminar el instrumento o no.
	10	Si el actor Usuario (ACT-001) selecciona no, el caso de uso finaliza.
	11	Si el instrumento que va a ser eliminado dispone de canales de entrada, para cada uno de ellos el sistema notifica al instrumento al que estuviera conectado el canal de entrada para que lo desconecte.
	12	Si el instrumento dispone de canales de salida, se pasa al flujo alternativo 12
	13	Si el instrumento está asociado a un actor Instrumento físico (ACT-002), el sistema elimina la asociación.
	14	El sistema elimina el instrumento de la lista de instrumentos creados
	15	El caso de uso finaliza
Postcondición	El instrumento seleccionado ha sido eliminado del sistema.	
Flujos alternativos	**Paso**	**Acción**

	12	12.1	Para cada instrumento en la lista *S*, el sistema busca qué canales entrada de ese instrumento están conectados al instrumento que va a ser eliminado.
		12.2	Para cada canal de entrada obtenido en el paso anterior, el sistema notifica al instrumento al que estuviera conectado el canal de entrada para que lo desconecte.
		12.3	El flujo alternativo termina y se pasa al punto 13 del flujo principal.
Comentarios	Ninguno		

Tabla 4.23: CU-002 Eliminar un instrumento

CU-003 Modificar un instrumento

CU-003	**Modificar un instrumento**	
Dependencias	**OBJ-001** Conexión y trabajo con instrumentos reales **OBJ-002** Emulación de instrumentos **OBJ-003** Configuración y síntesis de formas de onda	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera modificar el comportamiento de una unidad concreta de un instrumento	
Precondiciones	El sistema debe encontrarse iniciado	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) solicita al sistema la lista de instrumentos creados
	2	El sistema muestra todos los instrumentos creados al actor Usuario (ACT-001)
	3	El actor Usuario (ACT-001) selecciona el instrumento que desee modificar.
	4	El sistema muestra al actor Usuario (ACT-001) el instrumento seleccionado
		Punto de extensión: Crear forma de onda (CU-004)
		Punto de extensión: Cargar forma de onda (CU-005)
	5	El actor Usuario (ACT-001) selecciona aquella característica que desea modificar
	6	En caso de tener un número de alternativas limitado, el sistema muestra al usuario las opciones existentes para la característica seleccionada
	6	El actor Usuario (ACT-001) selecciona o introduce el valor que desea para esa característica
	8	El sistema almacena los cambios que ha producido el usuario
	9	Si el actor Usuario (ACT-001) no ha terminado de modificar todas las características que desea, se pasa al punto 5 del flujo principal
	10	El caso de uso finaliza
Postcondición	El instrumento ha guardado todas las modificaciones del usuario	
Comentarios	Ninguno	

Tabla 4.24: CU-003 Modificar un instrumento

CU-004 Crear forma de onda

CU-004	**Crear forma de onda**
Dependencias	**OBJ-003** Configuración y síntesis de formas de onda

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee agregar una onda simple o barrido a la forma de onda actual	
Precondiciones	El sistema debe encontrarse iniciado	
	El usuario está modificando un instrumento de tipo 'Generador de onda'	
Flujo normal	**Paso**	**Acción**
	1	El sistema muestra al actor Usuario (ACT-001), los tipos de onda diferentes que puede añadir
	2	El actor Usuario (ACT-001) elige el tipo de onda que desea agregar y selecciona 'Añadir'
	3	El sistema crea una nueva onda con los valores por defecto y la almacena en el sistema
	4	El caso de uso finaliza
Postcondición	Se ha creado y almacenado una nueva forma de onda con los valores por defecto	
Comentarios	Ninguno	

Tabla 4.25: CU-004 Crear forma de onda

CU-005 Cargar forma de onda

CU-005	**Cargar forma de onda**	
Dependencias	**OBJ-003** Configuración y síntesis de formas de onda	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee agregar un conjunto de ondas simples o barridos a la onda actual	
Precondiciones	El sistema debe encontrarse iniciado	
	El usuario está modificando un instrumento de tipo 'Generador de onda'	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona 'Cargar Onda'
	2	El sistema muestra al usuario una lista con los posibles definiciones de la forma de onda
	3	El actor Usuario (ACT-001) selecciona la definición de la forma de onda que desea
	4	El sistema genera y carga en el instrumento actual las formas de onda especificadas en la definición
	5	El sistema muestra al usuario las formas de onda añadidas
	6	El caso de uso finaliza
Postcondición	Se ha cargado en el instrumento una forma de onda a partir de la definición seleccionada	
Comentarios	Ninguno	

Tabla 4.26: CU-005 Cargar forma de onda

CU-006 Modificar forma de onda

CU-006	**Modificar forma de onda**	
Dependencias	**OBJ-003** Configuración y síntesis de formas de onda	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera cambiar las características de una onda simple o barrido de entre las actuales	
Precondiciones	El sistema debe encontrarse iniciado	
	El usuario está modificando un instrumento de tipo 'Generador de onda'	
Flujo normal	**Paso**	**Acción**
	1	El sistema muestra al actor Usuario (ACT-001) una lista con las formas de onda almacenadas y sus características
	2	El actor Usuario (ACT-001) selecciona la forma de onda que desea modificar
	3	El actor Usuario (ACT-001) selecciona la característica de la forma de onda que desea modificar
	4	Si la característica tiene un número de valores limitado, se pasa al flujo alternativo 4
	5	El actor Usuario (ACT-001) introduce o selecciona el nuevo valor de la característica que está modificando
	6	El sistema almacena el nuevo valor de la característica modificada
	7	En caso de que el actor Usuario (ACT-001) no haya terminado de modificar todas las características de la forma de onda seleccionada, se regresa al paso 3
	8	En caso de que el actor Usuario (ACT-001) no haya terminado de modificar todas las formas de onda que desea, se regresa al paso 1
	9	El caso de uso finaliza
Postcondición	Se han almacenado las modificaciones realizadas por el usuario	
Flujos alternativos	**Paso**	**Acción**
	4	4.1 El sistema muestra al usuario las posibles opciones que tiene la característica que se está modificando.
		4.2 El actor Usuario (ACT-001) selecciona la opción que desea de entre las posibles.
		4.3 El flujo alternativo termina y se pasa al punto 6 del flujo principal.
Comentarios	Ninguno	

Tabla 4.27: CU-006 Modificar forma de onda

CU-007 Ejecutar experimento

CU-007	**Ejecutar experimento**	
Dependencias	**OBJ-004** Metodología de trabajo	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera iniciar la simulación del experimento	
Precondiciones	El sistema debe encontrarse iniciado	
	El experimento se encuentra en estado de parada	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona 'Iniciar experimento'
	2	El sistema genera una lista con todos los instrumentos creados
	3	Para cada instrumento creado:
		3.1 El sistema notifica al instrumento para que se inicie
		3.2 Si el instrumento notificado está asociado a un actor Instrumento físico (ACT-002), se pasa al flujo alternativo 3
		3.3 Si quedan instrumentos en la lista sin notificar, se vuelve al punto 3.1
	4	El caso de uso finaliza
Postcondición	El experimento se encuentra en estado de ejecución	
Flujos alternativos	**Paso**	**Acción**
	3	3.1 El sistema envía al actor Instrumento físico (ACT-002) al que está asociado el instrumento actual la configuración que este tenga.
		3.2 El sistema envía al actor Instrumento físico (ACT-002) al que está asociado el instrumento actual la orden de iniciarse.
		3.3 El flujo alternativo termina y se pasa al punto 3.3 del flujo principal.
Comentarios	Ninguno	

Tabla 4.28: CU-007 Ejecutar experimento

CU-008 Parar experimento

CU-008	**Parar experimento**	
Dependencias	**OBJ-004** Metodología de trabajo	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera detener la simulación del experimento	
Precondiciones	El sistema debe encontrarse iniciado	
	El experimento se encuentra en estado de ejecución	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) selecciona 'Parar experimento'
	2	El sistema genera una lista con todos los instrumentos creados
	3	Para cada instrumento creado:

		3.1 El sistema notifica al instrumento para que se detenga	
		3.2 Si el instrumento notificado está asociado a un actor Instrumento físico (ACT-002), se pasa al flujo alternativo 3	
		3.3 Si quedan instrumentos en la lista sin notificar, se vuelve al punto 3.1	
	4	El caso de uso finaliza	
Postcondición	El experimento se encuentra en estado de parada		
Flujos alternativos	**Paso**	**Acción**	
	3	3.1	El sistema envía al actor Instrumento físico (ACT-002) al que está asociado el instrumento actual la orden de detenerse.
		3.2	El flujo alternativo termina y se pasa al punto 3.3 del flujo principal.
Comentarios	Ninguno		

Tabla 4.29: CU-008 Parar experimento

Capítulo 5
Diseño

"Todo lo que puedas imaginar es real."

Pablo Ruiz Picasso

5.1. Introducción

Una vez realizado el proceso de análisis, pasamos a la fase de diseño. Aquí transformaremos el resultado de la fase anterior, puramente conceptual (modelo del dominio), en un conjunto de clases dependientes del sistema y de la tecnología donde van a ser implementados; en este caso, en un PC con Windows 8.1 que disponga del framework .NET.

Generalmente es en este capítulo donde se expondría el diagrama de clases de diseño y los diagramas de paquetes, pero se ha optado por moverlos al siguiente capítulo. De esta manera, resulta más sencillo seguir la evolución de la aplicación a medida que han ido surgiendo nuevas necesidades.

5.2. Arquitectura lógica del sistema

La arquitectura lógica para esta aplicación es muy sencilla. En primer lugar disponemos de un ordenador tipo semitorre con el sistema operativo Windows 8.1 instalado. A su vez, dentro de Windows 8.1 está instalado el framework .NET versión 4. Nuestra aplicación se compilará para .NET, con lo cual hará uso tanto de sus bibliotecas como del intérprete CLR (de *Common Language Runtime*, entorno común de ejecución para lenguajes).

Nuestra aplicación hara uso de los drivers de Keysight para controlar el osciloscopio, a través de los componentes de IVI. Con ellos, el programa podrá comunicarse con el firmware del osciloscopio Keysight DSO-X 3104A, utilizando comandos GPIB a través del puerto USB. Algo similar sucede con el amplificador Lock-in, que ya que no dispone de puerto USB, ni la semitorre de GPIB, se comunicará con nuestra aplicación empleando la interfaz Keysight 82357B. De esta forma, podremos enviar los comandos GPIB encapsulados mediante paquetes USB, que la interfaz extraerá y mandará por el conector GPIB al Lock-in, y viceversa.

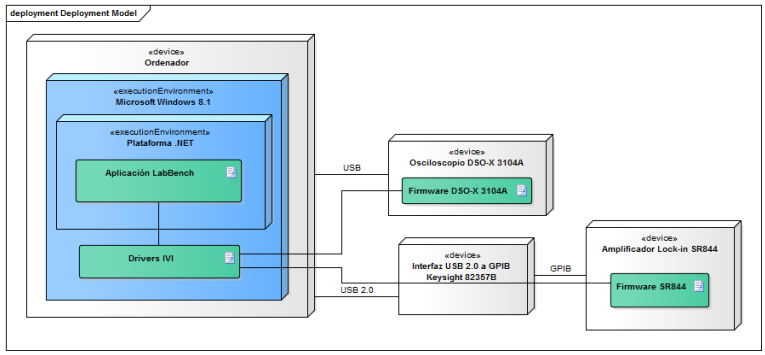


Figura 5.1: Diagrama de despliegue para la aplicación LabBench.

5.3. Descripción de los casos de uso de diseño

CUD-001 Crear un instrumento

CUD-001	**Crear un instrumento**	
Dependencias	**OBJ-002** Emulación de instrumentos	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera agregar una nueva unidad de un instrumento concreto al experimento	
Precondiciones	El sistema debe encontrarse iniciado	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) se dirige al formulario principal de la aplicación, despliega el menú principal, y pulsa con el botón izquierdo del ratón sobre el submenú 'Instrumento'
	2	El sistema despliega el submenú en el que muestra los tipos de instrumentos disponibles, 'Amplificador Lock-in', 'Generador de señal', 'Osciloscopio', y 'Registrador de datos'. A su vez, cada uno de estos abre un submenú en el que muestran las diferentes marcas disponibles para ese tipo de instrumentos, y cada marca despliega un submenú con los modelos disponibles para esa marca.
	3	El actor Usuario (ACT-001) navega por los submenúes y pulsa con el botón izquierdo del ratón sobre el instrumento que desee añadir.
	4	El sistema consulta en una lista interna cuantas instancias de ese tipo existen en memoria
	5	El sistema crea y registra una nuevo objeto del mismo tipo que el instrumento que ha seleccionado, es decir, AMPLIFICADOR LOCK-IN, GENERADOR DE SEÑAL, OSCILOSCOPIO, y REGISTRADOR DE DATOS

	6	El sistema asigna al atributo identificador un string único compuesto por una abreviatura del tipo de instrumento y el número de instrumento de ese tipo que está activo
	7	El sistema inicializa todos los *comboboxes* de los canales de entrada y salida y asignación a instrumento del nuevo objeto a 'Desconectado'
	7	El sistema crea un buffer cada canal de salida del instrumento y registra al objeto como escritor.
	8	El sistema pasa a primer plano la ventana asociada al objeto instrumento recién creado
	9	El caso de uso finaliza
Postcondición	Se ha creado una unidad de un instrumento del tipo especificado por el usuario	
Comentarios	Ninguno	

Tabla 5.1: CUD-001 Crear un instrumento

CUD-002 Eliminar un instrumento

CUD-002	**Eliminar un instrumento**	
Dependencias	**OBJ-001** Conexión y trabajo con instrumentos reales **OBJ-002** Emulación de instrumentos	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera eliminar una unidad concreta de un instrumento del experimento	
Precondiciones	El sistema debe encontrarse iniciado	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) busca en la barra de tareas, el nombre de la ventana que corresponde al instrumento que desea eliminar, y hace doble click sobre él.
	2	El sistema muestra al actor Usuario (ACT-001) la ventana asociada al instrumento con ese identificador
	3	El actor Usuario (ACT-001) cierra la ventana utilizando el aspa de la esquina derecha de la pantalla
	4	Si el instrumento dispone de canales de entrada, el sistema extrae de cada uno (obteniendo el identificador del primer *combobox* de cada canal de salida) los identificadores de los objetos a los que el actual está conectado, los resuelve a objetos, y los guarda en el vector E.
	5	Si el instrumento dispone de canales de salida, el sistema consulta la lista de objetos asociados como lectores al buffer al que el canal de salida está asociado como escritor, y genera con ellos el vector S.

	6		El sistema muestra al actor Usuario (ACT-001) un *msgbox* con el texto: 'Este instrumento está conectado a: ', a continuación la lista *E*, el texto 'Los siguientes instrumentos están conectados a este: ', luego la lista *S*, y el texto 'Los instrumentos anteriores quedarán desconectados. ¿Realmente desea eliminar el instrumento actual?'.
	7		El actor Usuario (ACT-001) decide si desea eliminar el instrumento o no.
	8		Si el actor Usuario (ACT-001) pulsa con el ratón sobre el botón 'No', el caso de uso finaliza. Si pulsa 'Sí', el caso de uso continúa.
	9		Si existen elementos en el vector *E* (si el instrumento que va a ser eliminado dispone de canales de entrada), el sistema elimina al objeto actual como lector del buffer al que estuviera asociado.
	10		Si el instrumento dispone de canales de salida, se pasa al flujo alternativo 10
	11		Si el objeto actual está asociado a un instrumento físico, el sistema notifica al objeto que lo controla para que elimine el puntero hacia el objeto instrumento a destruir.
	12		El sistema elimina el instrumento del vector de objetos tipo instrumento existentes.
	13		El caso de uso finaliza
Postcondición	El instrumento seleccionado ha sido eliminado del sistema.		
Flujos alternativos	**Paso**	**Acción**	
	12	12.1	Para cada instrumento en la lista *S*, el sistema verifica, para cada uno de sus canales de entrada, si éstos están apuntando al objeto instrumento a eliminar.
		12.2	Si están apuntando, el sistema elimina el puntero y pone los *combobox* del canal de entrada que apuntaba al objeto a borrar, como 'Desconectado'.
		12.3	El flujo alternativo termina y se pasa al punto 13 del flujo principal.
Comentarios	Ninguno		

Tabla 5.2: CUD-002 Eliminar un instrumento

CUD-003 Modificar un instrumento

CUD-003	**Modificar un instrumento**
Dependencias	**OBJ-001** Conexión y trabajo con instrumentos reales **OBJ-002** Emulación de instrumentos **OBJ-003** Configuración y síntesis de formas de onda

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera modificar el comportamiento de una unidad concreta de un instrumento	
Precondiciones	El sistema debe encontrarse iniciado	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) busca en la barra de tareas, el nombre de la ventana que corresponde al instrumento que desea eliminar, y hace doble click sobre él.
	2	El sistema muestra al actor Usuario (ACT-001) la ventana asociada al instrumento con ese identificador
		Punto de extensión: Crear forma de onda (CUD-004)
		Punto de extensión: Cargar forma de onda (CUD-005)
	5	El actor Usuario (ACT-001) selecciona aquella característica que desea modificar
	6	En caso de que la característica tenga un número de alternativas limitado, el sistema muestra, mediante un *combobox* las opciones existentes para ella
	6	El actor Usuario (ACT-001) selecciona la opción, en caso de ser un *combmbox* o introduce el valor, si se trata de un *textbox* que desea para esa característica.
	8	El sistema almacena los cambios que ha producido el usuario en el objeto. En algunos casos, serán almacenadas usando el objeto MY.SETTINGS.
	9	Si el actor Usuario (ACT-001) no ha terminado de modificar todas las características que desea, se pasa al punto 5 del flujo principal
	10	El caso de uso finaliza
Postcondición	El instrumento ha guardado todas las modificaciones del usuario	
Comentarios	Ninguno	

Tabla 5.3: CUD-003 Modificar un instrumento

CUD-004 Crear forma de onda

CUD-004	**Crear forma de onda**	
Dependencias	**OBJ-003** Configuración y síntesis de formas de onda	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee agregar una onda simple o barrido a la forma de onda actual	
Precondiciones	El sistema debe encontrarse iniciado	
	El usuario está modificando un instrumento de tipo 'Generador de onda'	
Flujo normal	**Paso**	**Acción**

	1	El sistema muestra al actor Usuario (ACT-001) dos tablas, una para agregar ondas simples, otra, debajo, para agregar barridos. Debajo de cada tabla se ubican los botones 'Cargar' y 'Guardar', a la izquierda, y 'Añadir' y 'Eliminar', a la derecha.
	2	El actor Usuario (ACT-001) decide si desea agregar una onda simple o un barrido a la forma de onda actual y pulsa sobre el botón 'Añadir' de la tabla correspondiente.
	3	El sistema genera un nuevo objeto ONDASIMPLE, o BARRIDO, dependiendo de la opción, y lo inicializa con los valores por defecto.
	4	El sistema añade una nueva fila a la tabla adecuada con los valores de ese objeto recién creado.
	5	El sistema agrega el nuevo objeto creado al objeto ONDACOMPUESTA.
	6	El caso de uso finaliza
Postcondición	Se ha creado y almacenado una nueva forma de onda con los valores por defecto	
Comentarios	Ninguno	

Tabla 5.4: CUD-004 Crear forma de onda

CUD-005 Cargar forma de onda

CUD-005	**Cargar forma de onda**	
Dependencias	**OBJ-003** Configuración y síntesis de formas de onda	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) desee agregar un conjunto de ondas simples o barridos a la onda actual	
Precondiciones	El sistema debe encontrarse iniciado	
	El usuario está modificando un instrumento de tipo 'Generador de onda'	
Flujo normal	**Paso**	**Acción**
	1	El sistema muestra al actor Usuario (ACT-001) dos tablas, una para agregar ondas simples, otra, debajo, para agregar barridos. Debajo de cada tabla se ubican los botones 'Cargar' y 'Guardar', a la izquierda, y 'Añadir' y 'Eliminar', a la derecha.
	2	El actor Usuario (ACT-001) decide si desea agregar una onda simple o un barrido a la forma de onda actual y pulsa sobre el botón 'Cargar' bajo la tabla correspondiente.
	3	El sistema abre un formulario estándar de selección de archivo.
	4	El actor Usuario (ACT-001) navega por los directorios hasta encontrar el archivo que desea y pulsa sobre el botón 'Abrir' del formulario estándar.

	5	El sistema abre el archivo e intenta decodificar su estructura XML extrayendo de él las formas de onda simples y los barridos genera y carga en el instrumento actual las formas de onda especificadas en la definición
	6	Para cada definición de onda simple o barrido, el sistema genera un nuevo objeto ONDASIMPLE o BARRIDO, y lo carga con los valores especificados en el XML.
	7	El sistema agrega una entrada en la tabla correspondiente con los valores de los objetos creados.
	8	El caso de uso finaliza
Postcondición	Se ha cargado en el instrumento una forma de onda a partir de la definición seleccionada	
Comentarios	Ninguno	

Tabla 5.5: CUD-005 Cargar forma de onda

CUD-006 Modificar forma de onda

CUD-006	**Modificar forma de onda**	
Dependencias	**OBJ-003** Configuración y síntesis de formas de onda	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera cambiar las características de una onda simple o barrido de entre las actuales	
Precondiciones	El sistema debe encontrarse iniciado	
	El usuario está modificando un instrumento de tipo 'Generador de onda'	
Flujo normal	**Paso**	**Acción**
	1	El sistema muestra al actor Usuario (ACT-001) dos tablas, una para agregar ondas simples, otra, debajo, para agregar barridos. Debajo de cada tabla se ubican los botones 'Cargar' y 'Guardar', a la izquierda, y 'Añadir' y 'Eliminar', a la derecha.
	2	El actor Usuario (ACT-001) decide si desea modificar la característica de una onda simple o barrido, y realiza un *click* simple sobre la fila que contiene la definición de la onda que desea modificar
	3	El actor Usuario (ACT-001) hace doble *click* sobre la celda que contiene la característica de la onda que desea modificar.
	4	Si la característica tiene un número de valores limitado, se pasa al flujo alternativo 4
	5	El actor Usuario (ACT-001) introduce por teclado en el *textbox* o selecciona el nuevo valor, a través del *combobox*, de la característica que está modificando
	6	El sistema actualiza el objeto asociado a la fila que ha sido modificada con el nuevo valor de la característica

	7		En caso de que el actor Usuario (ACT-001) no haya terminado de modificar todas las características de la forma de onda seleccionada, se regresa al paso 3
	8		En caso de que el actor Usuario (ACT-001) no haya terminado de modificar todas las formas de onda que desea, se regresa al paso 1
	9		El caso de uso finaliza
Postcondición	Se han almacenado las modificaciones realizadas por el usuario		
Flujos alternativos	**Paso**	**Acción**	
	4	4.1	El sistema despliega el menú del *combobox* con las posibles opciones que tiene la característica que se está modificando.
		4.2	El actor Usuario (ACT-001) selecciona la opción que desea de entre las posibles mediante un *click* simple.
		4.3	El flujo alternativo termina y se pasa al punto 6 del flujo principal.
Comentarios	Ninguno		

Tabla 5.6: CUD-006 Modificar forma de onda

CUD-007 Ejecutar experimento

CUD-007	**Ejecutar experimento**	
Dependencias	**OBJ-004** Metodología de trabajo	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera iniciar la simulación del experimento	
Precondiciones	El sistema debe encontrarse iniciado	
	El experimento se encuentra en estado de parada	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) se dirige al formulario principal de la aplicación, despliega el menú principal, y pulsa con el botón izquierdo del ratón sobre el submenú 'Simulación'
	2	El sistema despliega el submenú en el que muestra las acciones disponibles para la simulación, 'Iniciar', 'Pausar', y 'Parar'. Dado que el sistema se halla en estado de parada, los dos últimos elementos de este menú, Pausar', y 'Parar', se encontrarán desactivados.
	3	El actor Usuario (ACT-001) pulsa sobre el elemento 'Iniciar'
	4	El sistema accede internamente al vector I de objetos tipo IINSTRUMENTO creados hasta entonces.
	5	Para cada objeto I_n en el vector I:
		5.1 El sistema llama la función iniciar() del objeto I_n para que inicie sus funciones

		5.2 Si el instrumento notificado está conectado mediante un controlador a un actor Instrumento físico (ACT-002), se pasa al flujo alternativo 5	
		5.3 Si quedan objetos en la lista sin notificar, se vuelve al punto 5 con el siguiente elemento de la lista I, I_{n+1}.1	
	6	El caso de uso finaliza	
Postcondición	El experimento se encuentra en estado de ejecución		
Flujos alternativos	**Paso**	**Acción**	
	5	5.1	El sistema envía al actor Instrumento físico (ACT-002) al que está asociado el instrumento actual la configuración que este tenga, por ejemplo, el vector de valores de voltaje resultado de renderizar la forma de onda compuesta en caso de tratarse de un Generador de onda.
		5.2	El sistema envía al actor Instrumento físico (ACT-002) al que está asociado el instrumento actual la orden de iniciarse a través del controlador.
		5.3	El flujo alternativo termina y se pasa al punto 5.3 del flujo principal.
Comentarios	Ninguno		

Tabla 5.7: CUD-007 Ejecutar experimento

CUD-008 Parar experimento

CUD-008	**Parar experimento**	
Dependencias	**OBJ-004** Metodología de trabajo	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor Usuario (ACT-001) quiera detener la simulación del experimento	
Precondiciones	El sistema debe encontrarse iniciado	
	El experimento se encuentra en estado de ejecución	
Flujo normal	**Paso**	**Acción**
	1	El actor Usuario (ACT-001) se dirige al formulario principal de la aplicación, despliega el menú principal, y pulsa con el botón izquierdo del ratón sobre el submenú 'Simulación'
	2	El sistema despliega el submenú en el que muestra las acciones disponibles para la simulación, 'Iniciar', 'Pausar', y 'Parar'. Dado que el sistema se halla en estado de parada, los dos primeros elementos de este menú, Pausar', y 'Parar', se encontrarán desactivados.
	3	El actor Usuario (ACT-001) pulsa sobre el elemento 'Parar'
	4	El sistema accede internamente al vector I de objetos tipo IINSTRUMENTO creados hasta entonces.
	5	Para cada objeto I_n en el vector I:

		5.1 El sistema llama la función parar() del objeto I_n para que inicie sus funciones	
		5.2 Si el instrumento notificado está conectado mediante un controlador a un actor Instrumento físico (ACT-002), se pasa al flujo alternativo 5	
		5.3 Si quedan objetos en la lista sin notificar, se vuelve al punto 5 con el siguiente elemento de la lista I, I_{n+1}.1	
	6	El caso de uso finaliza	
Postcondición	El experimento se encuentra en estado de parada		
Flujos alternativos	**Paso**	**Acción**	
	5	5.1	El sistema envía al actor Instrumento físico (ACT-002) al que está asociado el instrumento actual la orden de detenerse a través del controlador.
		5.2	El flujo alternativo termina y se pasa al punto 5.3 del flujo principal.
Comentarios	Ninguno		

Tabla 5.8: CUD-008 Parar experimento

5.4. Diseño de la interfaz

En esta sección se explican los principios que se han tenido en cuenta a la hora de diseñar las pantallas que conforman la aplicación, y se presentan los esquemas que se implementarán en la siguiente fase. Para ello, se han utilizado técnicas y herramientas aprendidas en la asignatura 'Ingeniería de la interacción', del Máster en ingeniería informática.

Cuando se diseña una interfaz, conviene tener en cuenta la experiencia que los usuarios de la futura aplicación tienen con su actual entorno. Por ese motivo, el sistema está pensado como si el usuario estuviera trabajando en un laboratorio físico: se dispone de un armario con varios instrumentos, de diferentes marcas, modelos y características; se sacan los que sean necesarios para el experimento que se va a realizar, y se conectan entre sí con cables. Cuando está todo listo, se inicia el experimento, y en un momento dado, este concluye. Dado que estamos desarrollando una simulación digital, resulta interesante incluso tener la opción de pausar el proceso para reanudarlo más adelante en el mismo punto. Es posible encontrar ejemplos de este paradigma en muchas suites de simulación electrónica, como por ejemplo en Proteus.

Las líneas rojas identificadas en esta aplicación son tres, y han sido puestas a prueba con diferentes personas para ver si el diseño resultaba adecuado para los usuarios objetivo:

- Creación y añadido de instrumentos al experimento,
- Conexión y configuración de los instrumentos e
- Inicio y parada de la simulación

5.4.1. Dinámica de la aplicación y pantalla principal

La aplicación está compuesta por varios formularios diferentes. La ventana principal que aparece al abrir la aplicación simplemente contiene un menú y una imagen de fondo, de temática relacionada con la electrónica. Esta ventana controla el funcionamiento global de la aplicación, la adición de instrumentos y la simulación. Por ello, el menú principal tiene tres submenúes diferentes:'Instrumentos', 'Simulación' y 'Ayuda'. Empezaremos por el de instrumentos, que puede verse en la figura 5.2.

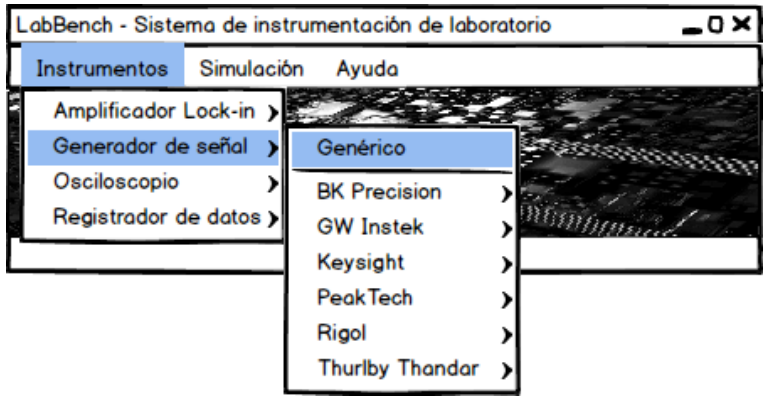


Figura 5.2: Mockup del formulario principal, submenú de instrumentos.

La idea detrás del submenú instrumentos es que éstos están clasificados, primero por su tipo (osciloscopios, generadores de ondas, amplificadores Lock-in, etc.), luego por su marca, y a continuación por el modelo. De cada tipo de instrumento existe un modelo genérico que presenta la funcionalidad común a esta clase de instrumentos, y que puede conectarse con un dispositivo hardware de ese tipo para realizar operaciones básicas. Para una funcionalidad extendida, conviene navegar por estos menúes hasta encontrar el dispositivo de la misma marca y modelo que el que dispongamos físicamente y conectarles.

Dentro de la aplicación también están considerados instrumentos que no pueden conectarse a un ordenador (porque no tienen puerto para ello), pero cuyo funcionamiento puede simularse igualmente. Un ejemplo de estos, como veremos más adelante, es el generador de funciones TG210 de Thurlby Thandar, Instruments, del que sólo se simula su funcionamiento porque no puede controlarse desde un ordenador.

Igual que en el símil del armario de instrumentos, es común (sobre todo en laboratorios docentes) tener varias unidades de los mismos modelos de un instrumento. Esto se ha tenido en cuenta en esta aplicación; es posible lanzar varias instancias del mismo instrumento volviendo a navegar por los menúes y volviendo a pinchar en el mismo instumento. Dicho de otro modo, cada vez que pinchamos en un instrumento del menú es como si agregáramos otro nuevo al experimento. Cada instancia de un instrumento (es decir, cada instrumento físico) está representado por una ventana en la aplicación, de forma que sea posible visualizar varios de ellos simultáneamente, por ejemplo dos osciloscopios y un Lock-in. Para esta clase de experimentos, en los que sea conveniente vigilar varios de ellos a la vez, puede ser recomendable disponer de varias pantallas conectadas al ordenador en configuración de escritorio extendido.

Pasemos ahora al submenú de simulación. Como se aprecia el las figuras 5.3 y 5.4, este submenú únicamente contiene tres elementos; 'Iniciar', 'Pausar' y 'Parar'. Al ejecutar la aplicación, el estado de la simulación es el de parada, y tanto para facilitar al usuario el conocimiento del estado actual en la máquina de estados del simulador, como para evitar posibles errores, la única opción habilitada es la de 'Iniciar', como puede verse en la figura 5.3. Eso significa que las otras dos opciones se muestran en gris y pinchar sobre ellas con el ratón no tiene ningún efecto (es decir, están deshabilitadas), 'Pausar' porque la simulación no se ha iniciado todavía y 'Parar' porque ya está parada y no tendría sentido volver a hacer click.

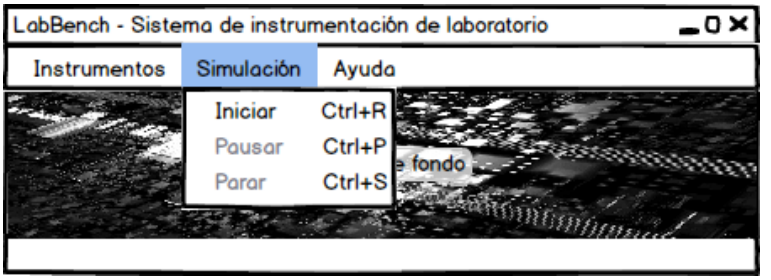


Figura 5.3: Mockup del formulario principal, submenú de simulación, con la simulación aún no inciada (es decir, parada).

Una vez iniciada la simulación, las dos opciones deshabilitadas se desbloquean, y la opción 'Iniciar' pasa a estar deshabilitada, como puede verse en la figura 5.4. Un añadido que puede resultar muy interesante para clarificar los estados y dado que un investigador, técnico u operario debería estar más atento a los datos del experimento que al estado de la aplicación es que, en lugar de tener que abrir el menú de simulación para conocer el estado de la misma, mostrar además un icono en la barra de notificaciones de la aplicación; de la misma forma que un reproductor multimedia, los iconos del triángulo, cuadrado y doble barra para indicar simulación activa, pausada y parada, respectivamente. Mostrando las tres con la función de deshabilitar algunas dependiendo del estado puede ahorrar algo de tiempo al funcionar como un acceso rápido. Esta barra de notificaciones es la que se encuentra en la zona inferior de las ventanas, y para todas las figuras de este apartado, está en blanco.

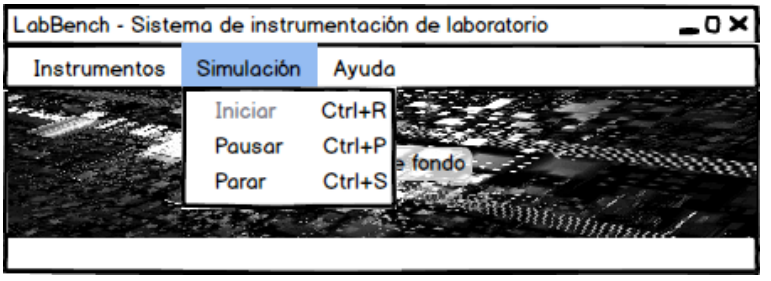


Figura 5.4: Mockup del formulario principal, submenú de simulación, con la simulación iniciada.

Por último, el submenú de ayuda sólamente contiene dos opciones; 'Temas de ayuda', que dispararía una aplicación similar a WinHelp en Windows XP y Windows Vista o un archivo PDF con el manual de la aplicación, y 'Acerca de...', que mostraría un formulario con la información de la aplica-

ción, logotipo, nombre, número de versión y de compilación, desarrolladores, fecha de compilación, etc. Este submenú puede verse en la figura 5.5.

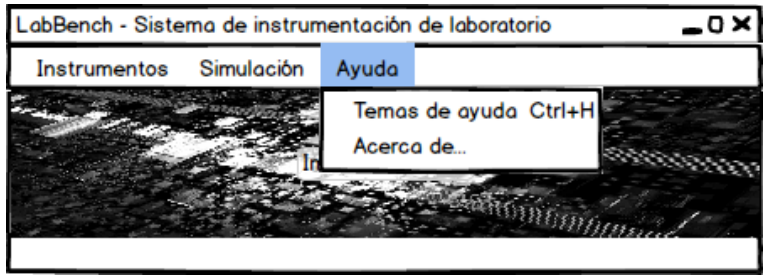


Figura 5.5: Mockup del formulario principal, submenú de ayuda.

5.4.2. Instrumentos

En este apartado podremos ver cómo se han adaptado los paneles de los distintos instrumentos, de qué forma se gestiona la existencia de instrumentos físicos e instrumentos simulados y cómo se ha adaptado el símil físico a la aplicación.

Como hemos comentado anteriormente, resulta una buena idea adaptar la interfaz de los programas al entorno que los usuarios ya están acostumbrados a utilizar. De esta forma nos aseguramos que la curva de aprendizaje sea lo más corta posible, con lo que serán capaces de manejar la aplicación con comodidad en muy poco tiempo.

A la hora de realizar las conexiones entre los diferentes instrumentos se ha pensado en un esquema tipo 'productor-consumidor'. Generalmente, cuando se realiza un conexionado, existe un dispositivo que produce los voltajes y otros los leen o modifican. Por ejemplo, el generador de voltajes 'produce' los voltajes, el osciloscopio los 'lee', y el Lock-in 'lee' valores de dos canales (referencia y salida del sistema bajo pruebas) y 'produce' los valores de salida en los canales de fase, cuadratura, desfase y módulo. Aunque físicamente es posible conectar una fuente de voltaje en CC a una fuente en CA para obtener una señal alterna con una componente de continua, esa opción no está disponible en este diseño, aunque puede ser fácilmente modificable para que así sea.

De esta forma, cada dispositivo, además de marca, modelo y tipo, tiene un nombre, que es único entre todos los instrumentos. De esta forma, su nombre distingue unívocamente a una instancia de un instrumento entre todas las demás, sean del mismo o de diferente modelo o tipo. Inicialmente, el sistema genera para cada nueva instancia de un instrumento (esto es, se asigna al crear su objeto en memoria) un nombre por defecto diferente, dependiendo del tipo del instrumento. Por ejemplo, para los generadores de señales, empezaría por GeneraSeñal1, GeneraSeñal2, GeneraSeñal3, y así sucesivamente. Con los osciloscopios, es Osciloscopio1, Osciloscopio2, etc. y con los Lock-in empieza por AmplifLockIn1. Para mayor conveniencia, el nombre se expone en un cuadro de texto modificable en las ventanas de cada instrumento, de forma que pueda sustituirse el nombre que en ese momento tenga por uno que le permita al usuario recordar mejor la función que tiene ese instrumento, como si fuera una pegatina. Por ejemplo, si un osciloscopio está vigilando los voltajes de la tercera etapa de amplificación de un circuito, puede ser útil renombrarle como EtapaAmplificacion3, o algo similar.

La forma de representar una conexión con otro instrumento depende de si esa conexión se considera de entrada o de salida. Las conexiones de salida están identificadas por un nombre que es común a todos los instrumentos de un modelo, y este nombre no puede modificarse. Esto significa que todos las instancias de un generador de señal TTi TG210 tendrán las conexiones de salida 'AuxOut', 'MainOut600Ohm', y 'MainOut50Ohm'. Estas conexiones de salida pueden estar, o bien desconectadas, o bien conectadas, y en cada instrumento en el que existe una conexión de este tipo simpemente se muestra un *combobox* con dos opciones; una es Desconectado, y la otra, el nombre de la conexión. Cuando se selecciona el nombre de la conexión, esa salida está activada.

Los instrumentos que tengan conexiones de entrada, como por ejemplo un osciloscopio, en el que típicamente todas sus conexiones son de este tipo, lo que se hace es referenciar conexiones de salida de otros instrumentos. ¿Cómo se logra esto? Utilizando dos *comboboxes* en cada conexión de entrada. Con el primero se selecciona el nombre único de un instrumento que tenga conexiones de salida (por ejemplo, GeneraSeñal1), y con el segundo, se selecciona el nombre del canal de salida al que queremos conectarnos de entre todos los de salida que exponga el modelo de instrumento al que pertenezca el que hemos seleccionado por nombre en el primer *combobox*.

Generador de señal

Como puede verse en la figura 2.3, uno de los generadores de funciones con los que trabajan los investigadores del laboratorio 1L012 es con el TG210 de Thurlby Thandar Instruments, que puede verse ampliado en 5.6. Transformar esta fotografía de su panel frontal en el formulario de una aplicación es una tarea sencilla, aunque no puede extraerse directamente debido a las limitaciones inherentes al uso del ratón. El resultado es el que puede verse en la figura 5.7.

Figura 5.6: Panel frontal de un generador de señal TTi TG210.

Uno de los cambios que ha sido necesario hacer respecto del panel del instrumento es reemplazar los potenciómetros por controles deslizantes (*sliders*). Cuando se emplea un ratón resulta más incomodo realizar movimientos circulares que en línea recta. Los *sliders* 'DC Offset', 'Symmetry' y 'Amplitud' ofrecen la misma funcionalidad que sus homólogos (aunque en el *mockup* faltan las dos imágenes de simetría que aparecen a izquierda y derecha del potenciómetro). Al igual que en el

dispositivo físico, el orden de magnitud o multiplicador de la frecuencia se controla por la botonera de la parte superior izquierda, y al activar un botón, el que ya estuviera pulsado previamente pierde el realce y el pulsado queda realzado hasta que se presione sobre otro. Es posible todos queden sin realzar pulsando sobre el que ya está realzado.

Ya que el panel frontal del instrumento presenta una gran densidad de indicaciones en torno al potenciómetro selector de frecuencia de la parte izquierda, en el diseño se ha optado por utilizar una caja de texto incrementable y decrementable para un ajuste más fino que utilizando el *slider*, donde podrían no caber todos los números. Otra solución posible podría pasar por haber alargado el control deslizante hasta que todos los números tuvieran cabida en la parte superior, pero esto implicaría que la ventana tuviera mayor anchura para conservar las proporciones. Al igual que en el panel físico, en la aplicación no aparece indicación alguna del orden de magnitud al que se refiere el valor del *slider* o de la caja de texto, y tanto en el caso del aparato físico como el virtual, es necesario un osciloscopio para poder conocer exactamente tanto el offset como la amplitud que se hayan seleccionado.

En cuanto a las conexiones, como se ha explicado anteriormente, este instrumento tiene tres conexiones de salida ('Aux Out', 'Main Out' con una impedancia de 600Ω, y otro 'Main Out', pero con una impedancia de 50Ω) y una de entrada ('Sweep In'). Sin embargo, existe un error en el *mockup* de la figura 5.7. Debajo de 'Sweep In' debería haber dos *comboboxes* en vez de uno, el primero para seleccionar el nombre único del instrumento a cuyo canal de salida se va a conectar y el segundo para elegir a qué canal de salida en concreto de los que presenta el instrumento se va a hacer la conexión.

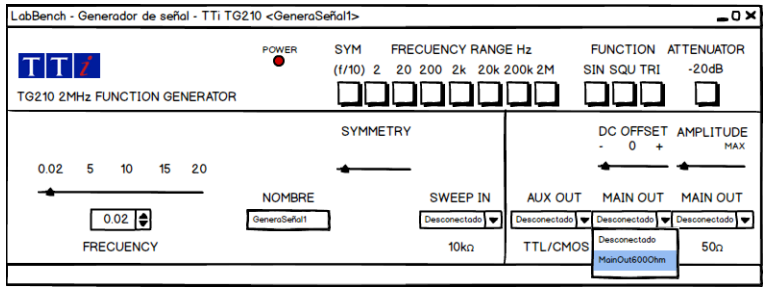


Figura 5.7: Mockup del generador de señales TG210, de Thurlby Thandar Instruments. Debajo de 'Sweep In' debería haber dos *comboboxes* en vez de uno.

También se ha diseñado el formulario de un generador de señal genérico, que puede verse en la figura 5.8. Existen varios grupos de controles (*groupboxes*). El primero contiene la caja de texto modificable que especifica el nombre de esa instancia de un generador de señal genérico y, en caso de existir uno o más generadores de señal conectados al PC, la caja de opción múltiple (*combobox*) permite seleccionarlos por marca y modelo. El *combobox* a su derecha permite seleccionar qué canal en concreto debe exportar la señal configurada en esta pantalla, en caso de que el instrumento disponga de varios (si sólo dispone de uno, se seleccionará automáticamente, aunque el control estará activo, pero sólo mostrará una opción, evidentemente).

El siguiente *groupbox*, 'Salida' permite conectar y desconectar la única salida que tienen los generadores de señal genérico, que se denomina 'ChannelOut'. Para cumplir con el requisito de que la aplicación deberá soportar tanto ondas simples como complejas y barridos, los dos siguientes *groupboxes*, 'Ondas simples' y 'Barridos' permiten especificar un conjunto de ondas simples y barridos,

respectivamente, que formarán la onda de salida.

En el *groupbox* 'Ondas simples', los botones 'Añadir' y 'Eliminar' permiten añadir una fila al final de la tabla, o eliminarla respectivamente. Al añadirla, todos los valores, menos el de número de onda (que es asignado automáticamente), aparecerán a cero (frecuencia de 0 Hz, amplitud de 0V, desfase de 0° y *offset* DC de 0 V). De esta forma, en caso de que el usuario pulse inadvertidamente el botón 'Añadir', la onda simple creada no tendrá efecto alguno. Para ajustar los valores, el usuario debe hacer doble *click* sobre la celda que desee modificar y escribir el nuevo valor que tendrá ese parámetro en la onda simple que se está modificando. Como hemos visto anteriormente, las ondas simples tienen cuatro parámetros, que son frecuencia, amplitud, desfase, y desplazamiento u *offset* DC. La casilla que indica el número de onda no es modificable, así que el doble click no tiene efecto alguno sobre las celdas de esta columna. El botón 'Eliminar' elimina la fila seleccionada (se seleccionan con un *click* simple del ratón), así que el sistema recalcula los valores de las casillas de la columna número de onda, que es equivalente a decir número de fila. De esta forma, como es posible introducir varias ondas sencillas, se crea en conjunto una forma de onda compuesta y de esta forma se cumplen dos de los tres puntos del requisito. Si se desea generar una onda simple, basta con que la tabla de ondas simples contenga un único elemento y la de barridos esté vacía.

En cuanto al botón 'Cargar', se ha visto anteriormente que tanto la clase ONDA como ONDACOMPUESTA implementan la interfaz definida en IXMLSERIALIZABLE. Esto permite que tanto los objetos de tipo ONDA como ONDACOMPUESTA puedan ser almacenados en ficheros XML en el disco duro. Este botón permite abrir un formulario estándar tipo selección de archivos para elegir un XML con la descripción de una onda compuesta (que puede contener una única onda simple) y rellenar la tabla con los datos de ese archivo. Un botón que falta en la ventana y que puede resultar de mucha utilidad es el del proceso opuesto, es decir 'Guardar', para salvar la tabla de ondas simples como archivo XML.

Para el *groupbox* 'Barridos', el funcionamiento es análogo. Cada barrido tiene una frecuencia inicial, una frecuencia final, una amplitud, que es constante durante todo el barrido, un desfase, en grados, el *offset* en DC de la señal, los ciclos por frecuencia y el salto de frecuencia. Dado que se trata de un barrido, esto es, de una señal cuya frecuencia se incrementa o decrementa con el tiempo, es necesario saber, cuántos ciclos se deben generar en cada frecuencia diferente, y cuantos Hz de diferencia hay entre una frecuencia y otra. Dicho de otra manera, un barrido supone generar k ciclos a la frecuencia inicial f_i, k ciclos a la frecuencia $f_i + d$, k ciclos a la frecuencia $f_i + 2d$, etc., donde k son los ciclos por frecuencia y d es el salto de frecuencia.

Además de estos parámetros, existe uno especial llamado 'Modo', que determina cómo se comporta un barrido en relación al resto de la onda. Para ver esto mejor, supongamos que el barrido 3 ya ha terminado completamente; ha llegado a la frecuencia final y ha hecho sus últimos ciclos. Pero imaginemos que existen otros barridos que aún no han terminado. En este caso ¿qué debería hacer la que ya lo ha hecho? Este comportamiento está regido por el parámetro modo'Modo', y tiene tres opciones. Una primera opción es la 'Mantener', que como su nombre indica, el barrido que termine antes que otros con esta opción mantiene la frecuencia final hasta que todos hayan terminado. Una segunda opción es la de 'Estirar'. Cuando un barrido esta ajustado como 'Estirar', los parámetros de ciclos por frecuencia y salto de frecuencia se ajustan automáticamente para que la duración del barrido sea igual a la del más largo, de tal forma que terminen a la vez. Una última opción, que no aparece en la figura 5.8, es la de 'Finalizar'. Si un barrido acaba antes que los demás, a partir de ese

momento desaparece de la señal de salida del generador.

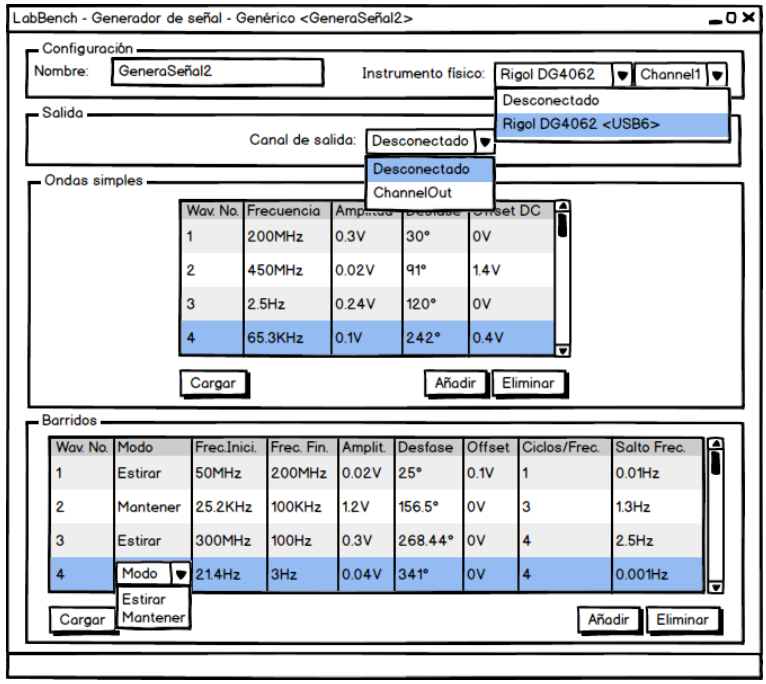


Figura 5.8: Mockup del generador de señal genérico.

Osciloscopio

El diseño de las pantallas de los osciloscopios genéricos es el siguiente, y está basada en la funcionalidad que ofrece un osciloscopio analógico básico.

En primer lugar y como todos los instrumentos, tiene un *groupbox* llamado configuración, que contiene únicamente el cuadro de texto donde se muestra y modifica el nombre de la instancia del instrumento. Debajo de él está el grupo de controles del *trigger*, donde se puede ajustar el nivel del mismo, en porcentaje respecto al área vertical de la pantalla, y el número de canal, referido a la fila de la tabla de canales que se encuentra debajo de este *groupbox*, que se debe utilizar como señal de referencia para el *trigger*. En caso de que el lector lo desconozca, un *trigger* sirve para definir un nivel de voltaje. El osciloscopio toma como referencia de tiempo ($t = 0$) y empieza a mostrar desde el lado izquierdo de la pantalla, cada vez que la señal de referencia supera ese nivel de voltaje. Esto ayuda tanto a estabilizar la imagen de la pantalla como a capturar eventos, como por ejemplo la transmisión de paquetes de datos por un bus, o un pulso.

El *groupbox* 'Horizontal' (que en la figura 5.9 aparece como 'Vertical' debido a una errata) sirve para definir el equivalente a la componente vertical de la relación de aspecto en un televisor, permitiendo ajustar a cuántos segundos equivale la diferencia entre cada una de las rayas verticales de los cuadrados blancos que aparecen en la 'pantalla' del osciloscopio como tal, que aparece a la izquierda de la ventana. La opción 'Retardo' se utiliza en conjunción con el *trigger* y permite definir que la

señal empiece a mostrarse un cierto tiempo (en el caso de nuestra aplicación, fijado en segundos), después de que el *trigger* se haya disparado.

La tabla inferior define el número de canales que se visualizan en la pantalla de osciloscopio de la izquierda de la ventana. El botón 'Añadir' añade una fila con los valores por defecto ('Source' en 'Desconectado', 'Channel name' en 'Desconectado', 'Volts/Div' en 0V, 'Offset' en 0V y 'Color' en transparente, de forma que la ventana no quede modificada en caso de que el usuario pulse el botón por error, como en el caso de los botones 'Añadir' de la figura 5.8) al final de la tabla. El botón 'Eliminar' borra la fila que esté seleccionada, que se hace con un *click* simple del ratón. Igual que en el caso del generador de señal genérico, para editar los datos de un canal basta con hacer doble click sobre una celda para editarla, excepto en la columna de número de canal que no es editable. Cuando se pulsa dos veces sobre una celda de la columna 'Source' aparecen los nombres de las instancias de todos los aparatos que dispongan de canales de salida, sean físicos o virtuales. En cuanto a las celdas de la columna 'Color', al pulsar dos veces sobre ellas aparece un selector de color que permite elegir el color que tendrá el canal en la pantalla de osciloscopio de la izquierda.

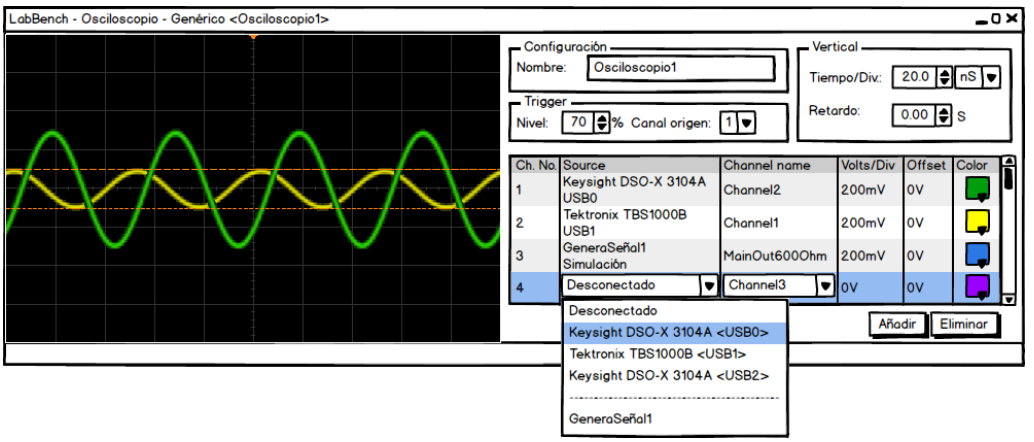


Figura 5.9: Mockup del osciloscopio genérico.

Registrador de datos

El registrador de datos es un instrumento que existe en el mundo físico y casi nunca se encuentra como tal, sino que comúnmente está embebido o asociado a otro. Por ejemplo, algunos osciloscopios pueden escribir los valores de voltaje en un fichero de una memoria USB. También se venden unidades que son una combinación de un sensor de temperatura, y una memoria USB. Para el caso que ocupa esta aplicación y con el objetivo de encapsular las distintas funcionalidades, se ha considerado como si se tratara de un tipo de instrumento diferente. La idea de este instrumento es la de 'leer' valores de una o varias conexiones diferentes a la vez, y escribirlas en un fichero en un cierto formato, generalmente CSV, pero en cualquier caso este diseño asume que todos los formatos de salida utilizan codifican los datos en ASCII, y no como valores binarios.

El primer *groupbox*, 'Configuración', contiene como en el resto de los casos, la caja de texto en la que puede verse y cambiarse el nombre de la instancia del instrumento; pero también la caja de texto

para el nombre, con la ruta completa, del archivo en el que se escribirán los datos, a cuya derecha hay un botón que permite lanzar el formulario estándar de selección de archivo. Un *combobox* permite cambiar el formato en el que se escribirán los datos, como por ejemplo CSV, XLS (Excel), etc. Dado que todos los voltajes se representan como valores decimales en precisión doble, un último *combobox* permite elegir si se desea que el separador decimal sea la coma o el punto. Esto es importante, ya que en algunas de las aplicaciones de prueba en Visual Studio, ha sido necesario modificar este parámetro para que el fichero resultante pudiera ser leído desde Matlab, lo que se debe a que Visual Studio detecta la configuración regional para detectar cuál de los dos actúa como separador decimal. Al escribir ficheros con una única columna en ASCII, y ya que la aplicación se ha compilado en un ordenador con configuración regional 'Español', utilizaba la coma como separador. Para que Matlab pudiera leer el vector sin problemas utilizando la función fscanf, era necesario reemplazar cada coma en el archivo por un punto, lo cual supone un engorro para los usuarios.

Por último, el *groupbox*, 'Estructura' contiene una tabla que permite definir a qué canal corresponde cada columna del archivo resultante. La primera fila de la tabla define el origen de datos de la primera columna del archivo, la segunda el segundo, y así sucesivamente. El funcionamiento es análogo al de la tabla de canales del osciloscopio genérico, con los botones 'Añadir' y 'Eliminar', la edición por doble *click*, el uso de los valores por defecto 'Desconectado' y 'Desconectado' en las columnas 'Origen' y 'Nombre canal', etc.

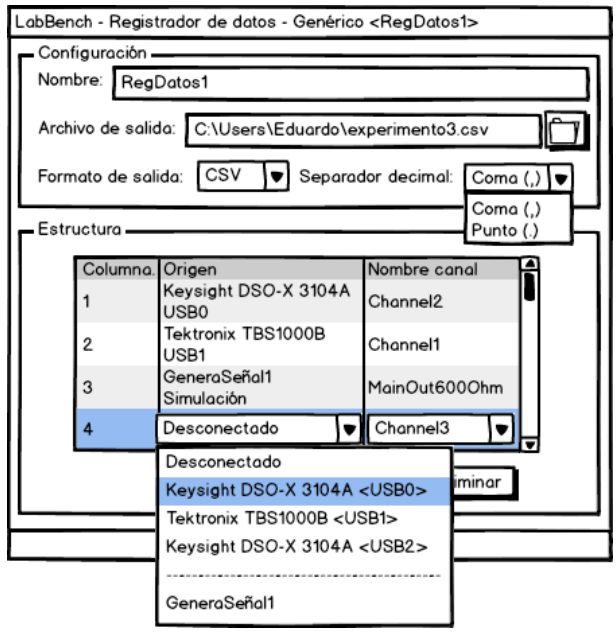


Figura 5.10: Mockup del registrador de datos genérico.

Amplificador Lock-in

Por último, respecto al Lock-in, se muestra el *groupbox* 'Configuración', donde como en el resto de los instrumentos aparece el cuadro de texto 'Nombre', y como en el generador de señal genérico,

aparecen los *comboboxes* para selección de instrumento físico, que muestran todos los amplificadores Lock-in conectados al ordenador. Como existen Lock-ins multicanal, el segundo *combobox* permite seleccionar el canal; con estos dos *widgets* se hace posible que esta ventana controle un amplificador Lock-in real, cumpliendo así con uno de los requisitos.

El *groupbox* 'Detección' permite al usuario configurar la ventana de tiempo en la que se detecta el máximo. Para análisis de señal *on line*, esto es, aquel en el que los datos llegan en forma de flujo continuo, no se pueden extraer todos los picos porque no se dispone de todos los datos. De tal forma que para el caso de un barrido o de una señal compuesta que estén generados por un dispositivo que no esté siendo controlado desde el sistema, pero estén siendo digitalizados (tanto la señal de referencia como la señal proveniente del sistema) mediante un osciloscopio, no tenemos información alguna de los instantes de tiempo en los que varía la frecuencia para poder tomarlo como ventana de tiempo en la que la onda que nos llega es una señal periódica estable (esto es, los picos que nos aparecen en esa ventana tienen la misma secuencia de voltajes en el mismo orden y aparecen con las mismas diferencias de tiempo entre ellos). Para ese caso, es necesario que el usuario marque una ventana de tiempo en la que buscar los máximos. ¿Para qué se utiliza esto? Hemos comentado previamente que un filtro pasa baja es equivalente a integrar la señal, o lo que es lo mismo, para eliminar la componente de alterna de la señal basta con promediar los valores de voltaje durante un ciclo completo. De esta manera, las desviaciones con respecto a la media producidas durante el semieje positivo se cancelarán con las producidas durante el negativo. Como el desfase varía con la frecuencia, la componente de continua variará al cambiar la frecuencia. Es por esto que necesitamos una ventana de tiempo en la cual tengamos valores estables. El valor de la ventana de detección es, en realidad, un *combobox* editable. El usuario puede escribir el valor que desee (con posibilidad de decimales) expresado en segundos, o desplegar el *combobox* y seleccionar la opción 'Auto'.

Esta opción sirve para el caso en el que el canal de referencia corresponda a un generador de ondas virtual, es decir, o bien simulado o controlado por el sistema. Ya que existe un patrón observador al que pueden suscribirse los dispositivos que estén utilizando el canal de salida del generador de ondas, es posible que éstos conozcan cuándo se producen variaciones en las frecuencias. Esto es útil sobre todo en el caso de los barridos de frecuencia. De esta forma, todo aquel que esté recibiendo datos de un generador de ondas puede conocer entre qué instantes de tiempo una frecuencia no ha variado y realizar el promediado en ese intervalo.

En el *groupbox* 'Entradas' se encuentran dos grupos de dos *comboboxes*, cada grupo para definir un canal de entrada; uno para la señal de referencia y otro para la señal de salida del sistema bajo test. Cualquier instrumento con al menos un canal de salida puede utilizarse como fuente de señal para cada entrada. Por último, el *groupbox* 'Salidas' permite conectar y desconectar cada uno de los cuatro canales de salida del instrumento; 'En fase', 'En cuadratura', 'Amplitud' y 'Desfase', todas referidas a características de la señal de salida del sistema bajo test. La aplicación muestra los valores de amplitud y desfase a medida que se van calculando y obteniendo.

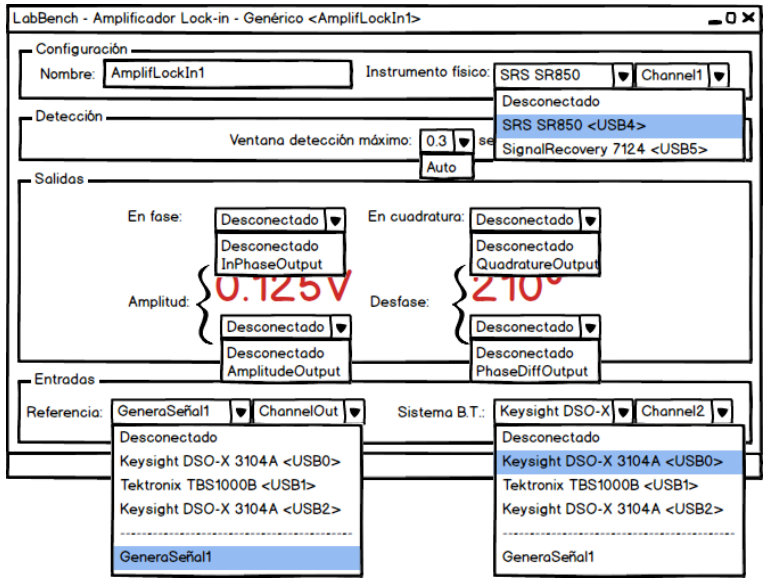


Figura 5.11: Mockup del amplificador Lock-in genérico.

Capítulo 6
Implementación, pruebas y resultados

"Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa."

Mahatma Gandhi

6.1. Introducción

Una vez realizado el análisis para extraer las clases que formarán el modelo de dominio, y complementar este modelo con las clases de diseño, el patrón arquitectónico y los diagramas de paquetes, podemos empezar a codificar la aplicación

6.2. Preparación del entorno

Inicialmente elegimos el lenguaje Matlab antes que C, C++ o C# porque habíamos programado con él recientemente en la asignatura de 'Métodos avanzados de razonamiento y representación del conocimiento', y por tanto nos resultaría más sencillo.

El paso resultó evidente, y fue instalar toda la suite de Matlab en el ordenador de desarrollo. Optamos por la versión R2013b porque disponíamos del DVD de instalación y de las licencias. Dado que éste es un proyecto que incluye procesado de señales y control de instrumentos, es necesario, en el paso en el que se pregunta sobre las bibliotecas que deberán instalar, marcar al menos las casillas correspondientes a 'Instrument Control Toolbox' y 'Signal Processing Toolbox'.

Lo siguiente que hicimos fue buscar e instalar los controladores para poder programar el osciloscopio desde el ordenador. Éstos pueden obtenerse y descargarse desde la página de Keysight [1], y se denominan '2000, 3000, 4000, 6000 InfiniiVision X-Series Oscilloscope IVI and MATLAB Instrument Drivers'.

Sin embargo, para poder instalarlos es requisito imprescindible tener instalados primero los 'IVI Shared Components', que es un conjunto de componentes software que contienen la funcionalidad

común a los controladores de todos los instrumentos que implementan los estándares IVI. Estos componentes pueden obtenerse desde la página oficial [37], y depende de si el sistema operativo del ordenador en el que se vaya a desarrollar. En nuestro caso se ha utilizado Windows 8.1 de 64 bits, de forma que el instalador que bajamos fue IVISharedComponents64_2.2.1.msi. La instalación se llevó a cabo utilizando los valores por defecto del instalador.

También es un requisito obligatorio para poder instalar los controladores del osciloscopio tener descargadas e instaladas las bibliotecas 'IO Libraries Suite', que pueden obtenerse desde la página de Keysight en [22]. Además de los controladores genéricos, este paquete también instala el programa 'Keysight Connection Expert', que, como más adelante veremos, permite obtener los identificadores de los dispositivos de Keysight conectados al ordenador. De la misma forma, añade un icono en el área de notificación de la barra de tareas, desde el que podemos acceder a la documentación, lanzar comandos y monitorizar los instrumentos, etc. La última versión del fichero que hemos podido probar se denomina IOLibSuite_17_0_19013.exe, y, de nuevo, se ha instalado con los valores por defecto.

Una vez satisfechos ambos requisitos, procedemos a instalar los drivers del osciloscopio. En nuestro caso, la última versión del instalador disponible se denomina driver_ivi_matlab_AgInfiniiVision_2_0_3_0_x64.msi. Una vez más, la instalación se realizó con los valores por defecto.

Para la realización del proyecto también hemos dispuesto de licencia para el software BenchLink Waveform Builder Pro, que se obtienen desde la página web de Keysight [3], accediendo a la pestaña 'Trials & Licenses' y descargando la versión de prueba, que posteriormente se activa con el número de serie de la licencia. Una vez instalado, basta con acceder al icono de la aplicación en la parte superior izquierda de la pantalla; se desplegará un menú, cuyo último elemento es 'Licensing'. Al abrir el submenú asociado a 'Licensing', pulsando sobre el último elemento, 'Install license', aparece una ventana donde podemos introducir el número de serie.

6.3. Primera iteración

La primera versión del software la realizamos, como ya hemos explicado previamente, utilizando Matlab, pero la información que encontrábamos que nos pudiera ser útil estaba reducida a código fuente de ejemplo que no nos funcionaba correctamente. Lo que sí encontramos fue la descripción de los comandos GPIB para el aparato. Resulta inicialmente un poco extraño que no disponiendo el aparato se controle por GPIB teniendo únicamente puerto USB. Como más adelante descubriríamos, las bibliotecas realmente son una serie de *wrappers*, que transforman las llamadas a sus funciones en comandos GPIB que luego mandan por transferencias USB al osciloscopio.

La primera referencia a un código que nos funcionó la encontramos en un PDF obtenido de internet, y permitía obtener la lista de osciloscopios conectados al ordenador. Posteriormente, descurbimos que existía un único ejemplo de prueba en código Matlab en la carpeta de los Componentes IVI, concretamente, en C:\Program Files\IVI Foundation\IVI\Drivers\AgInfiniiVision\Examples\MATLAB. Combinando y modificando los códigos anteriores, logramos configurar el osciloscopio para capturar de forma continua los valores medidos a través de los canales 1 y 2 (y por extensión al resto de los canales). Buscando en la página de la compañía que produce Matlab, MathWorks, encontramos código de ejemplo que permite obtener también datos de los canales del osciloscopio, incluso con una

interfaz gráfica [4].

Posteriormente, descubrimos que Matlab también dispone de sugerencias de programación tipo Intellisense de Visual Studio, y tratamos de configurar el generador de ondas para que generara una forma de onda creada por nosotros, sin suerte. Descubrimos que a pesar de que las llamadas se ejecutaban correctamente, el osciloscopio no se veía afectado, con lo que, tras múltiples pruebas, dedujimos que la función no estaba implementada en el *wrapper* que transforma las llamadas a funciones de Matlab en llamadas a funciones del driver IVI.

```
%Obtener datos del osciloscopio
%osc = oscilloscope ();
%targets = getResources (osc);
%targets
%set (osc, 'Resource', 'USB0::0x0957::0x17A0::my53XXXXX::0::INSTR');
%connect(osc);
%%mex -setup
%%connect(osc);
%data = getWaveform(osc);
%plot(data);
clc
%A partir de aqui cargamos el driver bueno
try

    %Create driver instance
    driver = instrument.driver.AgInfiniiVision();

    %Edit resource and options as needed.  Resource is ignored if option Simulate=true
    resourceDesc = 'USB0::0x0957::0x17A0::my53XXXXX::0::INSTR';
    initOptions = 'QueryInstrStatus=true, Simulate=true, DriverSetup= Model=, Trace=false';
    idquery = true;
    reset   = false;

    driver.Initialize(resourceDesc, idquery, reset, initOptions);
    disp('Driver Initialized');
    disp(blanks(1));

    %Comando para parar la medicion
    %driver.DeviceSpecific.Measurements.Initiate();
    %Llamamos al generador de funciones
    generador = driver.DeviceSpecific.WaveformGenerator;
    generador.Function = 'AgInfiniiVisionWaveformGeneratorFunctionSinusoid'
    generador.OffsetVoltage = -2.5;
    generador.Frequency = 2;
    generador.Amplitude = 5.0;
     whitebg('black');
    clear title xlabel ylabel
    hf=figure('position',[0 0 eps eps],'menubar','none');
  while (true)
   [WaveformArray, InitialX, XIncrement] = driver.DeviceSpecific.Measurements.Item('Channel1'↩
       ).ReadWaveform(inf);
   [WaveformArray2, InitialX, XIncrement] = driver.DeviceSpecific.Measurements.Item('Channel2↩
       ').ReadWaveform(inf);

   subplot(2,2,1);
   plot(WaveformArray,'y');
   clear title xlabel ylabel
   title = 'Canal 1';
   subplot(2,2,2);
   plot (WaveformArray2,'g');
   clear title xlabel ylabel
   title = 'Canal 2';
    if strcmp(get(hf,'currentcharacter'),'q')
        close(hf)
       break
   end
   drawnow
```

```
end
catch exception
    disp(getReport(exception));
end

if driver.Initialized
    driver.Close();
    disp('Driver Closed');
end

disp('Done');
disp(blanks(1)');
```

Listing 6.1: Código en Matlab resultante de la primera iteración. La extracción de datos de los canales funciona correctamente, no así el control del generador de funciones.

El código con el que finalizó esta iteración puede verse en el listado 6.1. Los primeros comandos comentados permiten extraer el conjunto de osciloscopios conectados al ordenador. En primer lugar se crea un objeto tipo OSCILLOSCOPE mediante la función homónima, que se guarda en la variable *osc*. Utilizando ese objeto, puede llamarse a la función getResources, que devuelve los objetos del tipo dado. A continuación se muestra el contenido del objeto TARGETS, para ver los identificadores de los posibles osciloscopios a los que conectarse.

Además de éste código, desarrollamos algunas funciones a mayores que nos permitieran sintetizar ondas sinusoidales con sólo una llamada, por ejemplo.

Los identificadores de los dispositivos son cadenas de caracteres del estilo del que aparece en las líneas 5 y 19, como por ejemplo *USB0::0x0957::0x17A0::my53XXXXX::0::INSTR*, para los dispositivos conectados al bus USB (el segundo campo es el identificador de vendedor, o VID, USB, el tercero el PID USB, el cuarto el número de serie del dispositivo, etc.); *GPIB0::07::INSTR* para dispositivos conectados al bus GPIB, y *TCPIP0::157.88.111.XXX::INSTR* si el instrumento puede ser controlado a través de la red. La información sensible de este apartado ha sido sustituída por 'X'.

6.4. Segunda iteración

Al inicio de esta fase del proyecto se descubrió que los componentes IVI también podían utilizarse desde código C#. Mi compañero Andrés sugirió cambiar a este lenguaje por la imposibilidad de continuar utilizando Matlab, y por haberlo utilizado en su Proyecto Fin de Carrera, como se ha comentado anteriormente.

Un primer código de ejemplo en este lenguaje está disponible desde C:\Program Files\IVI Foundation\IVI\Drivers\AgInfiniiVision\Examples\CSharp\CsExample1. De nuevo, sólo existe un proyecto, pero fue suficiente como para empezar a crear un esquema de clases básico y escribir código en él.

Uno de los proyectos de esta etapa tiene por nombre 'SimuladorOndas', cuya función era precisamente esa, definir una serie de frecuencias para poderlas mezclar y ver el resultado en el propio programa. La interfaz era muy simple, y puede verse en la figura 6.1. Mediante el botón 'Añadir onda', se añadía una nueva entrada al *listbox* de la izquierda, debajo de la pantalla del osciloscopio, con los detalles de los controles de la izquierda. Cuando se pulsa sobre uno de los elementos del *listbox*, sus valores se cargan a los controles de la izquierda. Si se escribe algo en ellos, existe un elemento del *listbox* seleccionado, y se pulsa el botón 'Modificar onda', el elemento seleccionado pasaba a

tener los nuevos valores definidos a la derecha. Tras cada acción de añadido, modificado o eliminado de una onda, la pantalla del osciloscopio de la parte superiore se actualizaba con los datos del nuevo vector resultado de renderizar la onda compleja.

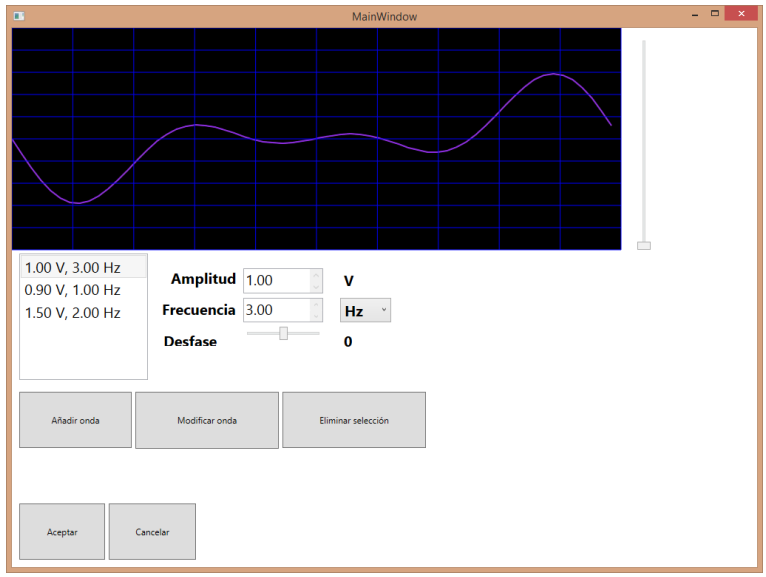


Figura 6.1: Aspecto del formulario principal del proyecto 'SimuladorOndas'.

El diagrama de clases está representado en la figura 6.2, que está formado, en la práctica, por cuatro clases: MAINWINDOW, SINEMIXER, y UTILIDADESGRAFICAS. La primera controla la interfaz gráfica, recibiendo llamadas a sus funciones al dispararse eventos gráficos, y también con los métodos de dibujo para el osciloscopio. La segunda clase se encarga de producir, a través de la función generateSineMix, que está sobrecargada, el vector de valores de voltaje resultado de renderizar las ondas simples definidas. Sin embargo, la forma con la que ésta función recibe los valores no es la adecuada, ya que sus parámetros tercero y cuarto son dos vectores, uno con los valores de las amplitudes de las ondas, y el segundo con sus frecuencias. La función toma el primer elemento de cada array para formar la primera onda simple, el segundo de cada una para formar la segunda onda, y así sucesivamente. Se vió que una solución mejor era encapsular las ondas como un objeto, con sus valores de amplitudes y voltajes, que se implementaría en la siguiente iteración.

La aplicación guarda la lista de ondas simples mediante el objeto MY.SETTINGS, que permite guardar la configuración de una aplicación a las aplicaciones desarrolladas desde .NET. De esta manera, la información de las distintas ondas se codificaba como una única cadena de caracteres que se salva al cerrar la aplicación, y se parsea para reconstruir los valores de las ondas al arrancarla.

Llegó un momento en el que el código era muy poco práctico, por tener el código responsable de generar y controlar la interfaz gráfica meszclado con el código que debería formar parte del modelo de dominio, y las clases no eran las adecuadas.

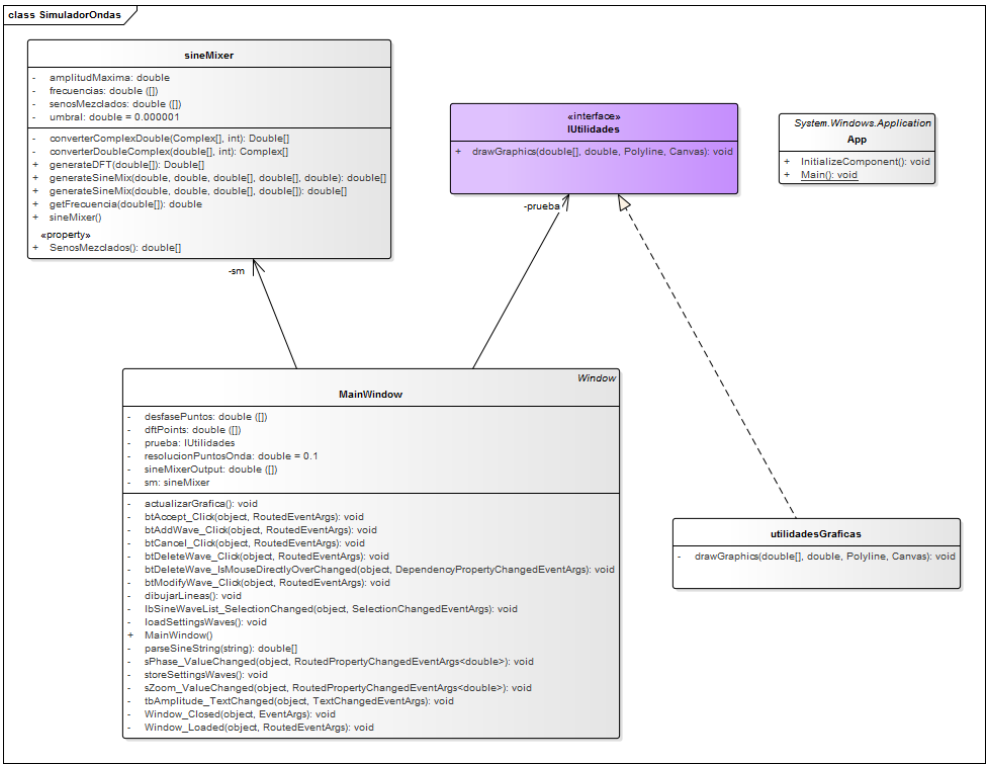


Figura 6.2: Diagrama de clases completo del proyecto 'SimuladorOndas'.

6.5. Tercera iteración

La tercera iteración del proyecto apareció con la refactorización del código de la segunda iteración. Para ello se creó un nuevo proyecto llamado, precisamente, 'ProyectoRefactorizado', en el que se rediseñó el esquema de clases. Posteriormente, y para evitar una nueva refactorización, se creó y desarrolló paralelamente el proyecto 'EsqueletoInicial', que contiene los cambios sobre 'ProyectoRefactorizado' que serán incorporados en la cuarta iteración, entre otros muchos.

6.5.1. Proyecto 'ProyectoRefactorizado'

En este proyecto, se rediseñó completamente el esquema de clases y se movió y refactorizó el código de la segunda iteración. Esta fue la primera ocasión en la que logramos realmente conectar con el osciloscopio, mandarle una forma de onda compuesta al DAC, y recibir los valores de voltaje de vuelta de los canales, como puede verse en la figura 6.3.

En ella, se puede ver parte del montaje experimental. El generador de ondas del osciloscopio, ajustado en modo 'Arbitrary Waveform', recibe una serie de valores de voltaje en punto flotante, que se envían al driver mediante dos posibles funciones, setData() y setDataDAC(). La primera necesita que los datos estén especificados en punto flotante entre 0 y 1, y están mapeados al rango de voltajes al que está limitado el generador de ondas, esto es, 0 son -2.5 V y 1 son 2.5 V. El segundo utiliza

números enteros. Una vez los datos salen del conector BNC situado en la parte inferior izquierda, se utiliza una 'T', uno de cuyos extremos se conecta al canal 1 del osciloscopio. De esta manera, es posible ver la señal que sale del generador de ondas, y como hemos comentado previamente, utilizarlo como canal de referencia. De la tercera toma de la 'T', sale un cable apantallado que pasa por el circuito bajo test, en este caso formado por un par de resistencias y un condensador. La señal sale del circuito como variación en la intensidad y pasa a un amplificador de transimpedancia, que transforma esa señal de intensidad en una señal de voltaje y además la amplifica. La salida del amplificador se conecta al canal 2 del osciloscopio, y se recibe por nuestro programa para ser procesada como señal de salida del sistema.

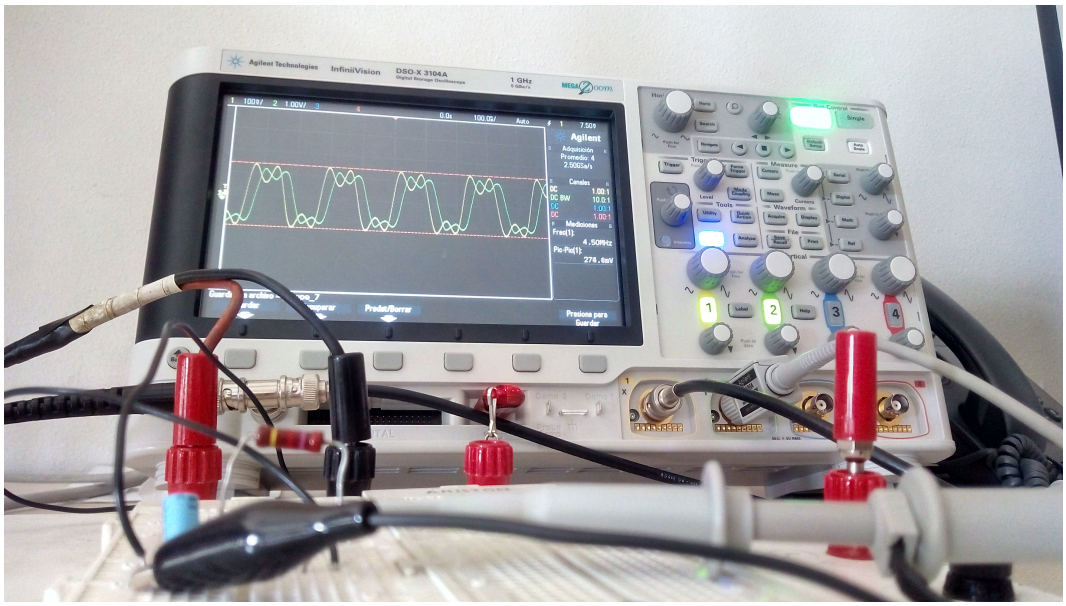


Figura 6.3: Montaje de prueba, en el que el DAC del osciloscopio DSO-X 3104A ha recibido una forma de onda compuesta generada por nuestra aplicación y el circuito de la protoboard la desfasa. *Imagen por cortesía de Francisco Javier Peña Rubio.*

Uno de los problemas que detectamos, como se verá más adelante, era que, a pesar de todos nuestros esfuerzos por multiplicar la señal de salida por algún factor, nunca obteníamos la misma amplitud generando una señal sinusoidal por programa, que sintetizándola directamente desde el menú del generador de ondas como tal, por unos pocos decimales. Intentamos, empleando un analizador de protocolo USB, visualizar los datos que el driver mandaba al osciloscopio para saber cómo codificaba los valores de voltaje. De hecho, llegamos a tratar de ponernos en contacto con la compañía y no obtuvimos respuesta.

En cuanto al diseño, la aplicación está realizada utilizando el patrón arquitectónico Modelo-Vista-Modelo de vista, abreviado MVVM, que es el que recomienda Microsoft para las aplicaciones desarrolladas en .NET. Las clases que forman parte del paquete del Modelo pueden verse en la figura 6.4.

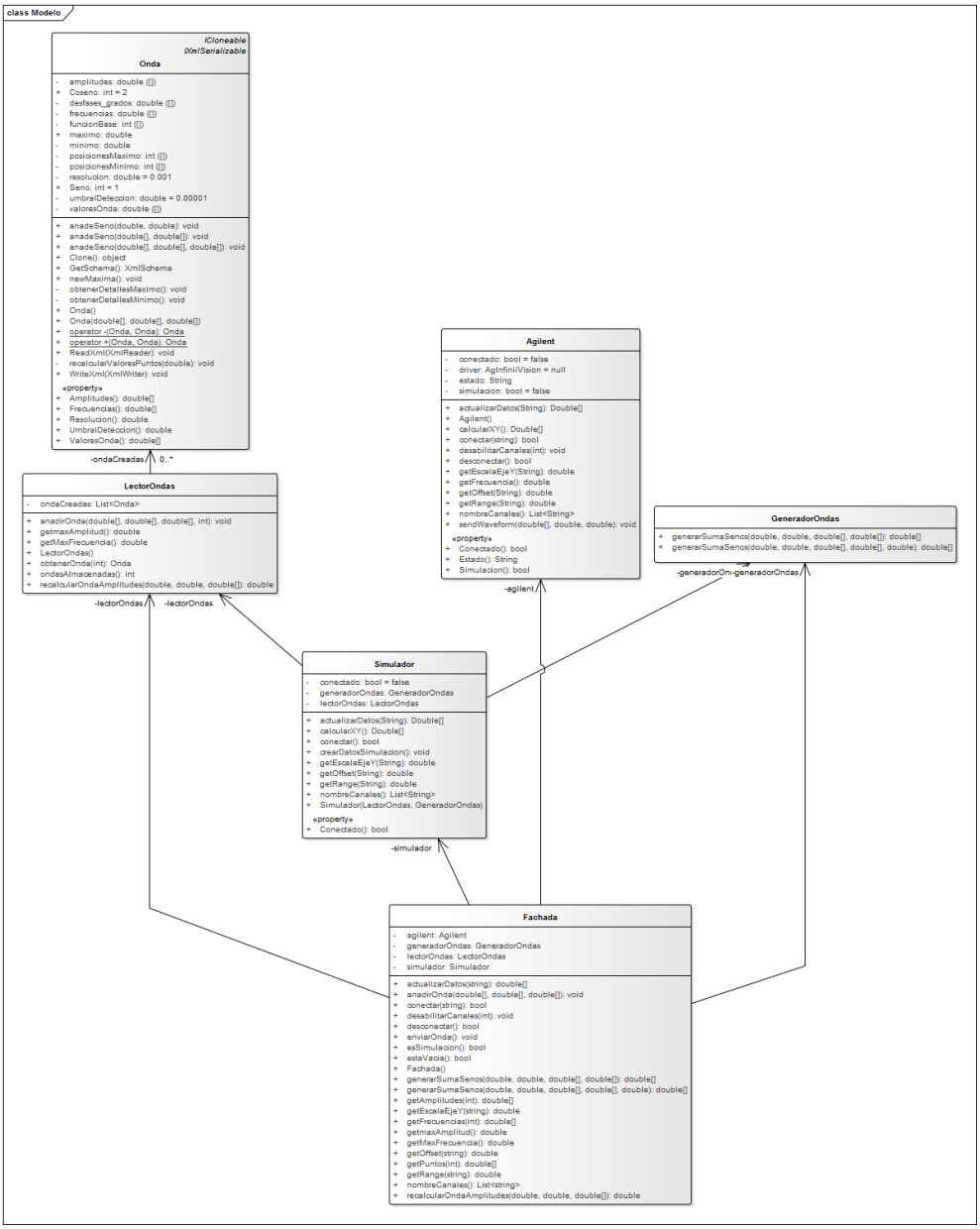


Figura 6.4: Diagrama de clases del paquete 'Modelo' del proyecto 'ProyectoRefactorizado'.

Interfaz gráfica

Este software dispone de dos pantallas; la pantalla principal, y la del mezclador de ondas. Desde la primera, que puede verse en la figura 6.5, es posible conectarse y desconectarse del osciloscopio físico, actualizar manualmente la señal que se muestra en la parte superior de la pantalla realizando una captura desde el osciloscopio (aunque en las últimas versiones el código es capaz de realizar

capturas de forma continua y actualizar la interfaz del programa automáticamente), configurar una onda mediante el mezclador de ondas, lo que abría el segundo formulario, y enviar al generador de ondas la forma de onda configurada previamente.

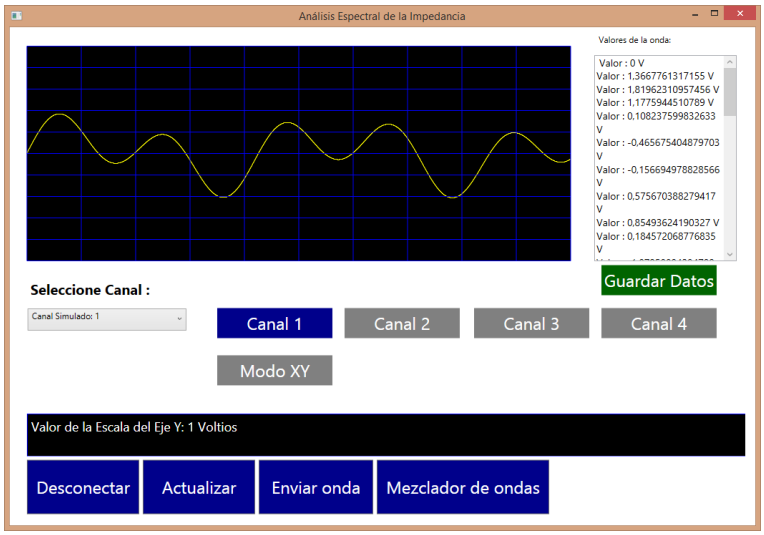


Figura 6.5: Aspecto del formulario principal del proyecto 'ProyectoRefactorizado'.

Desde el selector de canal es posible elegir de cuál de los cuatro canales debe extraerse la señal, y el que está actualmente en uso aparece resaltado en azul, el resto en gris. El programa también puede funcionar en modo simulación exactamente igual que si dispusiera de conexión al instrumento, renderizando la señal del mezclador de ondas y pasándola al osciloscopio de la aplicación en vez de al físico.

A la izquierda de la pantalla aparece un *listbox* con los valores del vector de voltajes que se han recibido del osciloscopio (o del mezclador de ondas en modo simulado), y almacenarlos en un fichero de texto. Gracias a ésto se han podido obtener las imágenes de la figura 6.7.

La segunda pantalla es la del mezclador de ondas, como hemos contado al principio de este apartado. Funciona de manera muy parecida a la aplicación 'SimuladorOndas' de la segunda iteración. Podemos verla en la figura 6.6.

Reconstrucción de ondas

Uno de los problemas con los que nos encontramos era la elevada cantidad de ruido con que llegan las señales del osciloscopio. Esto se puede apreciar mejor en la figura 6.7. Supone un inconveniente porque implica que los algoritmos de procesado no podrán asumir los valores que obtienen como ciertos. Por ejemplo, a la hora de detectar los picos, tiene sentido buscar el máximo de los valores del vector, y encontrar aquellos que se aproximen para obtener todos los picos. Sin embargo, siempre aparece algún valor espurio, que provoca que un pico sea más alto que el resto.

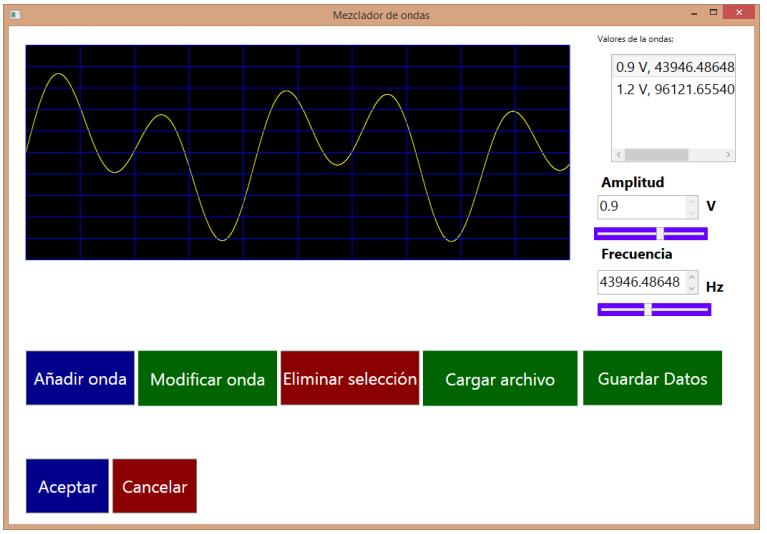


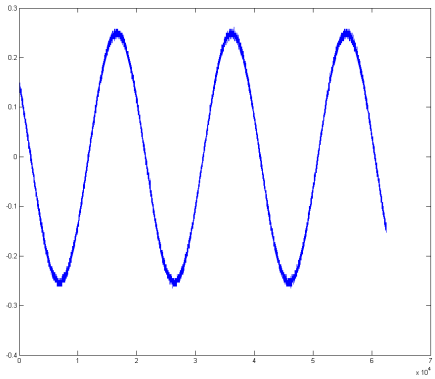
Figura 6.6: Aspecto del formulario del Mezclador de Ondas del proyecto 'ProyectoRefactorizado'.

Mirando detenidamente la imagen, parece evidente que el ruido es de carácter discreto, esto es, la señal parece estar escalonada y limitada únicamente a ciertos valores. Por este motivo supusimos que se trataba de un problema con el chip del DAC, concretamente en dos parámetros de diseño llamados No-Linealidad Integral (INL, *Integral Non-Linearity*) y No-Linealidad Diferencial (DNL, *Differential Non-Linearity*).

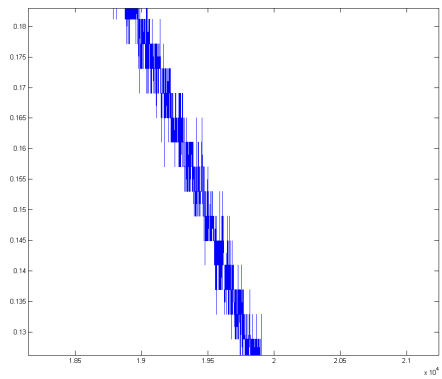
El primer algoritmo propuesto funcionaba para detectar picos siempre y cuando se aplicaran a señales de este tipo, con ruido 'escalonado', y un ejemplo gráfico se puede ver en la figura 6.8. Lo denominamos algoritmo del faro y consistía en recorrer el array realizando dos operaciones: la primera buscar el máximo, y la segunda, obtener la diferencia vertical mínima y promedio entre un punto y el consecutivo. Una vez obtenido el punto más alto, se le restaba el valor de la diferencia promedio multiplicada por 1.5, obteniendo el nivel marcado por la línea roja en la figura. De esta forma, se obtiene un límite inferior bastante robusto para detectar picos. A continuación, se recorre todo el array de nuevo. Cuando el valor de voltaje supera o baja del nivel marcado por el umbral, se carga la posición horizontal de ese punto en un vector V. Si ahora analizamos detenidamente el vector V, veremos que hay varios conjuntos de valores muy próximos que corresponden a los valores que han quedado por encima de la línea, pero que por efecto del ruido, suben y bajan. Entre estos conjuntos de valores existen saltos que corresponden a la diferencia entre el final de un pico y el final de otro. De esa forma podemos saber entre qué posiciones horizontales están los picos, y calcular sus puntos medios para hallar el máximo.

Reconstrucción de ondas

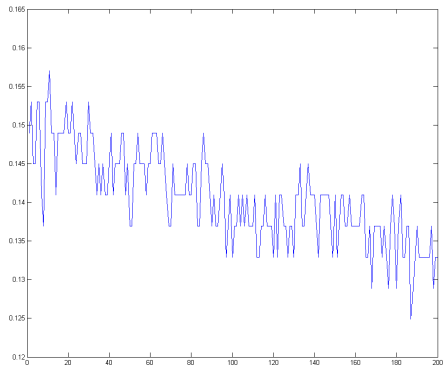
Otra estrategia que implementamos fue convolucionar el vector devuelto con lo que en procesado de imagen se conoce como máscaras, que en este caso se trata de vectores lineales. Hicimos pruebas con las de medias y desenfoque gaussiano, obteniendo mejores resultados con esta última, y encontrando que existía un punto a partir del cual ampliar el tamaño de la máscara dejaba de tener



(a) Sin zoom



(b) Aumentada en la segunda pendiente descente de 6.7a y limitada a unos pocos cientos de muestras



(c) Aumentada a la mitad de 6.7b, y limitada a unas decenas de muestras

Figura 6.7: Señal de 1V, 800Hz, tras ser enviada al DAC y recibida por la aplicación a través del

efecto.

Finalmente, el error desapareció al encontrar un modo del osciloscopio que realiza un promediado con una ventana de N=4, que corrige completamente el problema.

Introducción Ya sabemos que la fórmula que describe una onda sinusoidal es la siguiente

$$y(t) = A \cdot sin(\omega t + \theta) \tag{6.1}$$

donde A es la amplitud de la onda, ω es la velocidad angular, que es equivalente a $2\pi f$, t es el instante de tiempo para el que se calcula el valor del seno (que a su vez es la variable independiente de la ecuación, generalmente), θ es el desfase de la onda e y es el valor que toma la onda para un instante de tiempo t dado.

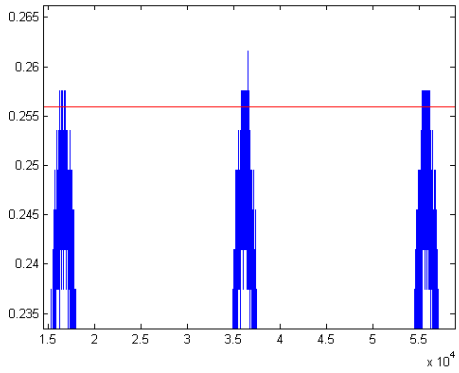


Figura 6.8: Funcionamiento del algoritmo del faro.

Amplitud Hay que recordar que la amplitud de una onda es la distancia que hay entre el punto medio que alcanza y el máximo valor que toma.

Cuando disponemos del vector de puntos que toma la onda, es decir, $y(t)$, podemos obtener fácilmente la amplitud de la onda si obtenemos el máximo valor que alcanza, menos el mínimo, dividido entre dos. Así, el código utilizado para calcular la amplitud de la onda sería:

```
public void extraer(double[] puntos)
   {
   double puntosMax = puntos.Max();
   double puntosMin = puntos.Min();
   Amplitud = ((puntosMax - puntosMin) / 2f);
   }
```

Desfase Teniendo ya calculada la amplitud A, y teniendo el vector de datos, disponemos del primer dato del vector, al que supondremos obtenido en $t = 0$. Por tanto, si volvemos a la Ecuación 6.8, sustituimos y despejamos:

$$y(0) = A \cdot sin(\omega 0 + \theta) \quad (6.2)$$

$$= A \cdot sin(\theta) \quad (6.3)$$

$$\frac{y(0)}{A} = sin(\theta) \quad (6.4)$$

$$sin^{-1}\left(\frac{y(0)}{A}\right) = \theta \quad (6.5)$$

y de esa forma habríamos obtenido el desfase.

Echando un vistazo a la documentación de la red Microsoft Developer Network (MSDN), puede verse que las funciones Math.Sin [26] y Math.Asin [25] aceptan y devuelven, respectivamente, los ángulos en radianes, y no en grados. Dado que nuestro sistema almacena los ángulos en grados, entre 0° y 360°, tenemos que convertirlos antes de pasarlos a las funciones anteriores utilizando las

fórmulas

$$\text{Grados} = \text{Radianes} \cdot \frac{180^\circ}{\pi} \tag{6.6}$$

$$\text{Radianes} = \text{Grados} \cdot \frac{\pi}{180^\circ} \tag{6.7}$$

De tal forma que el código quedaría de la siguiente manera:

```
public void extraer(double[] puntos)
   {
   //...
   Fase = Math.Asin(puntos[0] / Amplitud);
   Fase = Fase * (180f / Math.PI);
   if (Fase < 0)
      Fase += 360;
   }
public double[] renderizar(double resolucion, double ciclos)
   {
   double fase_radianes = Fase * (Math.PI / 180);
   for (int j = straight_array.GetLowerBound(0); j <= straight_array.GetUpperBound(0); j++)
      {
      sine_array[j] += Amplitud * Math.Sin(((2f * Math.PI * frecuencia_local) * ↩
           straight_array[j]) + fase_radianes);
      }
   }
```

Sin embargo, si realizamos algunas pruebas con el código obtenido, podemos ver que no siempre funciona correctamente. Concretamente, con cualquier ángulo comprendido entre 90° y 270° nos dan

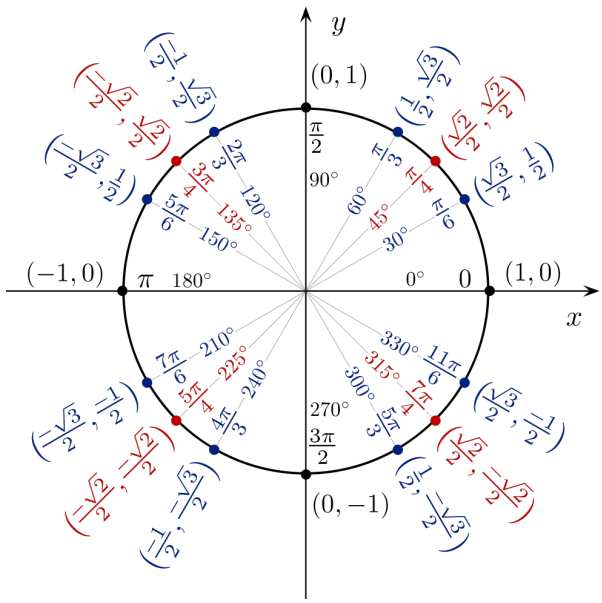


Figura 6.9: Círculo unidad, mostrando los ángulos en grados, radianes y coordenadas. Tomado de[39].

De tal forma que para poder distinguir si estamos en la "mitad izquierda" del círculo unidad (segundo y tercer cuadrantes) o en la "mitad derecha" (primero y cuarto), deberemos saber si, en el

momento inicial, la onda estaba ascendiendo o descendiendo. Podríamos ver si el siguiente punto toma un valor mayor, pero en un caso real, lo más posible es que la onda esté afectada por ruido.

Esto podría evitarse realizando un salto mayor, pero cabe la posibilidad de que, si ese salto no está lo suficientemente bien calculado, suponiendo la cercanía de un pico, este segundo punto esté en la zona descendente. O incluso en otro ciclo, dependiendo de la tasa de muestreo y de la frecuencia de la onda.

Una solución mejor consiste en detectar directamente el primer pico y el primer valle. Si el primer pico está más cerca del origen que el primer valle, estaremos ascendiendo, y en caso contrario (primer valle más cerca del origen), descendiendo.

Eso implica que también necesitaremos un detector de valles, que no suelen venir incorporados entre las funciones que ofrecen las bibliotecas ya que son de fácil implementación; basta con invertir la onda, calculando el valor máximo que toma, para a continuación restar cada punto de la onda a ese valor. A continuación se ejecuta el detector de picos sobre la onda invertida para obtener los puntos de los valles en la onda original.

Obtener la imagen especular de un vector respecto al eje de ordenadas, definido por $\pi/2$ y $3\pi/2$ (o, lo que es lo mismo, 90° y 270°) es bastante sencillo. Basta con restar, a cualquier ángulo que se encuentre entre $\pi/2$ y $3\pi/2$, el número π (180°) para obtener su correspondiente en la mitad que contiene al cero. Por ejemplo, y dado que generalmente es más sencillo pensar en grados que en radianes, la imagen especular respecto al eje de ordenadas de 150° sería $180° - 150° = 30°$.

Dado que nuestras clases trabajan con desfases entre 0° y 360°, como hemos dicho anteriormente, hemos de normalizar, ya que en caso de que se deseen invertir ángulos entre 180° y 270°, el resultado es negativo. Por ejemplo, $180° - 240° = -60°$.

```
public void extraer(double[] puntos)
   {
   //...
   //Calculamos los picos utilizando el filtro de Savitzky-Golay
   DoubleVector dv = new DoubleVector(puntos);
   PeakFinderSavitzkyGolay peak_finder = new PeakFinderSavitzkyGolay(dv, 5, 4);
   peak_finder.LocatePeaks();
   double maxptos = puntos.Max();
   //"Invertimos" la onda inicial
   double[] reversed_ptos = new double[puntos.Length];
   for (int i = puntos.GetLowerBound(0); i <= puntos.GetUpperBound(0); i++)
      reversed_ptos[i] = maxptos- puntos[i];
   //Obtenemos los valles, sacando los picos de la onda invertida
   DoubleVector dv2 = new DoubleVector(reversed_ptos);
   PeakFinderSavitzkyGolay valley_finder = new PeakFinderSavitzkyGolay(dv2, 5, 4);
   valley_finder.LocatePeaks();
   //Obtenemos la fase tal y como la devuelve el arcoseno
   Fase = Math.Asin(puntos[0] / Amplitud);
   //Si el primer pico esta mas cerca del primer dato que el primer valle
      //obtener la "imagen especular" del angulo respecto del eje pi/2 a 3pi/2
   if (peak_finder[0].X > valley_finder[0].X)
      Fase = Math.PI - Fase;
   Fase = Fase * (180f / Math.PI);
   if (Fase < 0)
      Fase += 360;
   }
```

Frecuencia La reconstrucción de la frecuencia, una vez detectados los picos, es sencilla pero nos sirvió para detectar una carencia de nuestro proyecto. Hasta ahora, tanto la función renderizar(), como extraer(), tomaban como argumento y devolvían, respectivamente, un vector de números en punto flotante de doble precisión. Pero con esos datos resulta imposible calcular la frecuencia, ya que, disponiendo sólo de los valores que toma el voltaje, no tenemos indicación alguna de en cuánto tiempo

se ha obtenido la muestra, necesario para saber cuánto tiempo transcurre entre picos, y a partir de ese valor, para calcular la frecuencia.

Por este motivo fue necesario añadir al sistema la clase MUESTRA, que simplemente contiene dos variables: un vector de números en punto flotante de doble precisión, y un número en punto flotante que indica el tiempo, en segundos, que ha sido necesario para tomar la muestra almacenada en el vector.

Ondas almacenables

La estructura de los archivos XML en los que se almacenan y de los que se recuperan las ondas es sencilla. En primer lugar, la primera línea del fichero, como en la mayoría de los archivos de este tipo, indica que está estructurado en XML y la codificación. Al menos con la versión del Framework .NET utilizado (4.5), los ficheros se escriben en UTF-8.

El resto del archivo contiene las definiciones de las ondas. La etiqueta <wave_container> marca, como su propio nombre indica, un contenedor de ondas, y es obligatoria. Dicho de otra manera, representa una onda compuesta. Dentro de esta etiqueta podemos encontrar etiquetas <wave>, que marcan la definición de una onda simple.

Cada una de las ondas simples debe contener las definiciones de tres de sus características, expresadas como números de doble precisión: la amplitud de la onda (<amplitude>), que se especifica en Voltios; su frecuencia (<amplitude>), en Hercios; y su fase (<phase>), en grados, y cuando es un fichero escrito por el software el valor de éste último campo se define entre 0 y 360.

Por tanto, un ejemplo de un fichero XML con esta estructura y conteniendo dos ondas simples, la primera de 1V, 800Hz y 0°, y la segunda de 4.2V, 6200Hz y 275°, sería el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<wave_container>
    <wave>
        <amplitude>1</amplitude>
        <frequency>800</frequency>
        <phase>0</phase>
    </wave>
    <wave>
        <amplitude>4.2</amplitude>
        <frequency>6200</frequency>
        <phase>275</phase>
    </wave>
</wave_container>
```

Pero ¿cómo puede hacerse para especificar una onda simple? La etiqueta <wave_container>, como ya hemos dicho, representa a un contenedor de ondas. La explicación que se ofrece a continuación fue la que motivó la creación del proyecto 'EsqueletoInicial', que veremos en el apartado 6.5.2.

Para esta iteración, a partir del fichero XML se genera una clase ONDA. Sin embargo, la estructura del fichero resulta ser más flexible que el diagrama de clases. Esto se debe a que al ser efectivamente

un contenedor, representa una onda compleja que puede tener cero o más ondas simples definidas dentro de él. El resultado no cambia, ya que si sólo existe una onda simple en el contenedor, la señal de salida será esa onda simple. Pero si existen dos o más será una onda compuesta. De esta forma podemos decir que una onda simple equivale a una onda compuesta, formada por una única onda simple.

De esta forma, la funcionalidad de la clase ONDA puede romperse de acuerdo al patrón compuesto, como veremos en la figura 6.11. Análogamente, las clases tipo ONDACOMPUESTA pueden contener cero o más ondas simples en 'EsqueletoInicial'. Más adelante veremos cómo distinguir si el archivo XML definía una única onda simple o una onda compleja.

Las ventajas de este diseño de contenedor son evidentes. Por ejemplo, ya que nuestros tutores nos solicitaron que los barridos de frecuencia estén soportados, podría definirse una etiqueta <chirp> dentro de <wave_container>, que a su vez contuviera un par de <wave> (uno para definir la amplitud y frecuencia iniciales y la otra para las finales), los ciclos por cada frecuencia y el salto de frecuencia. Esto deberá implementarse en la cuarta iteración, que contiene las clases para los barridos.

6.5.2. Proyecto 'EsqueletoInicial'

Dado que la clase Onda había crecido demasiado y debía descomponerse, se optó por crear un proyecto separado que permitiera tanto probar el buen funcionamiento de las nuevas clases como los algoritmos de Lock-in que harían uso de ellas. Otro buen motivo para ello es que las clases que dependían de la antigua versión de Onda estaban demasiado acopladas como para modificarlas sobre 'ProyectoRefactorizado', ya que eso supondría que el proyecto entero dejaría de compliar hasta que la nueva estructura de clases estuviera totalmente desarrollada y sus dependencias corregidas. De esta forma, podía probarse antes de ser integradas en 'ProyectoRefactorizado'.

El algoritmo de 'EsqueletoInicial' puede verse en la figura 6.10 y funciona para el caso estático (también llamado procesamiento *off line*), esto es, genera dos vectores de datos y a continuación les procesa. Como tiene todos los datos a la vez, puede encontrar el máximo real. En este diagrama se han utilizado líneas azules para denotar el proceso que sufre la señal de referencia, y naranjas para la de salida del sistema.

El algoritmo parte creando una ONDASIMPLE, a la que ajusta una frecuencia, una amplitud y una fase aleatoria. Esa onda será la onda de referencia. A continuación se clona el objeto tipo ONDASIMPLE anterior, y se le cambia la fase a otra generada también aleatoriamente. Esta será la onda de salida del sistema bajo test, que ha sido desfasada respecto de la de referencia por él. Posteriormente las dos ondas son renderizadas, transformando su especificación como amplitud, frecuencia y fase en un vector de datos que representa los valores de voltaje.

De los vectores con los valores de voltaje se pasa a crear dos nuevos objetos ONDASIMPLE, y a cargarles con esos vectores mediante la función extraer, que a partir de los vectores obtiene de nuevo la especificación dela onda en términos de amplitud, frecuencia y fase. ¿Qué sentido tiene transformarlos en vectores para luego hacer lo opuesto? De esta forma, estamos representado el mismo proceso que tendrá lugar en el sistema final, donde el generador de funciones transformará el objeto en valores de voltaje y se los enviará al DAC del osciloscopio. La onda pasará por el D.U.T., y el ADC del canal 1 obtendrá la onda de referencia, y el del canal 2 la onda de salida del D.U.T., pero los extraerá como un vector de valores de voltaje y así se los enviará al sistema, que debe reconstruir de nuevo las características de ambas ondas.

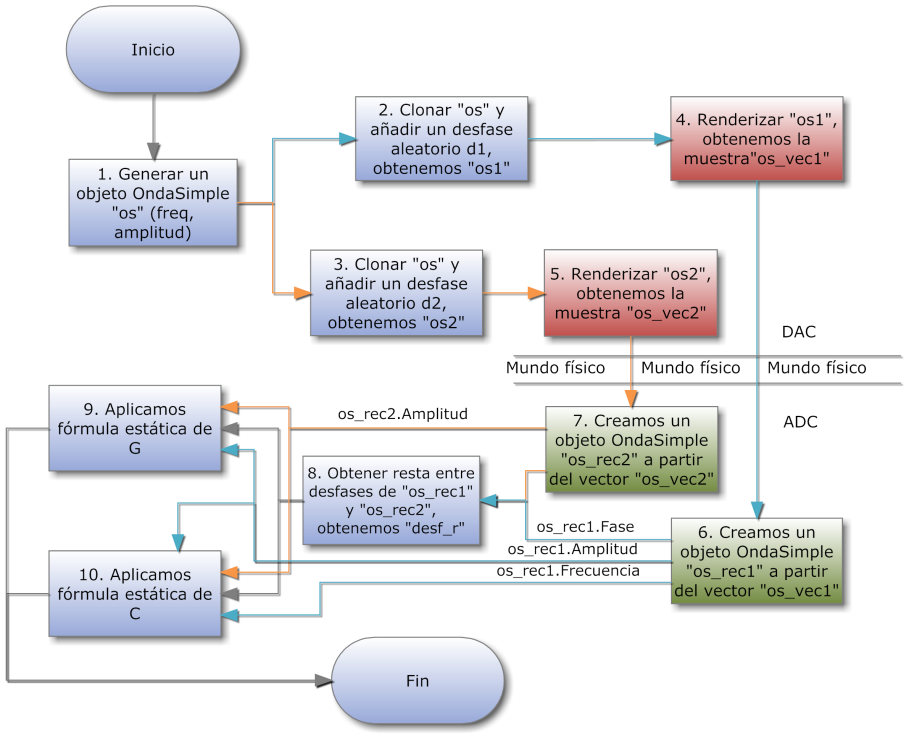


Figura 6.10: Diagrama de flujo del algorimo seguido por el programa 'EsqueletoInicial'.

El siguiente paso es aplicar las fórmulas de cálculo de la capacitancia que hemos descrito en el apartado 2.4.4, utilizando la amplitud y la fase de la señal de salida del sistema, y la amplitud, frecuencia de la señal de referencia.

Hemos visto en el apartado 2.4.3, y más concretamente en la figura 2.18, que, ya que el osciloscopio manda al ordenador el contenido del DAC, que es idéntico al que se puede visualizar por la pantalla, y que el propio instrumento, al utilizar la función AutoScale (incluso sin hacerlo), realiza un corte de la señal de referencia en cualquier punto, introduciendo un desfase aleatorio, es necesario obtener este desfase. Ya que la señal que obtenemos del D.U.T. tendrá el desfase introducido por éste más el introducido por el osciloscopio debido al corte, es necesario calcularle para restársele a las dos señales (referencia y sistema). Por eso al principio del algoritmo se le ha añadido un desfase aleatorio a la señal de referencia.

El diagrama de clases para gestión de ondas que utiliza este proyecto está reflejado en la figura 6.11.

En él podemos ver que existe una interfaz llamada IONDA, que define las características comunes a todos los tipos de ondas, sean simples o compuestas. Esto es, todas ellas deben poder convertir su definición en un vector de valores de voltaje (RENDERIZAR), así como el proceso inverso, pasar de un vector a las definiciones de las ondas (EXTRAER).

Respecto a los ficheros XML, en esta versión de la implentación estos ficheros deben leerse crean-

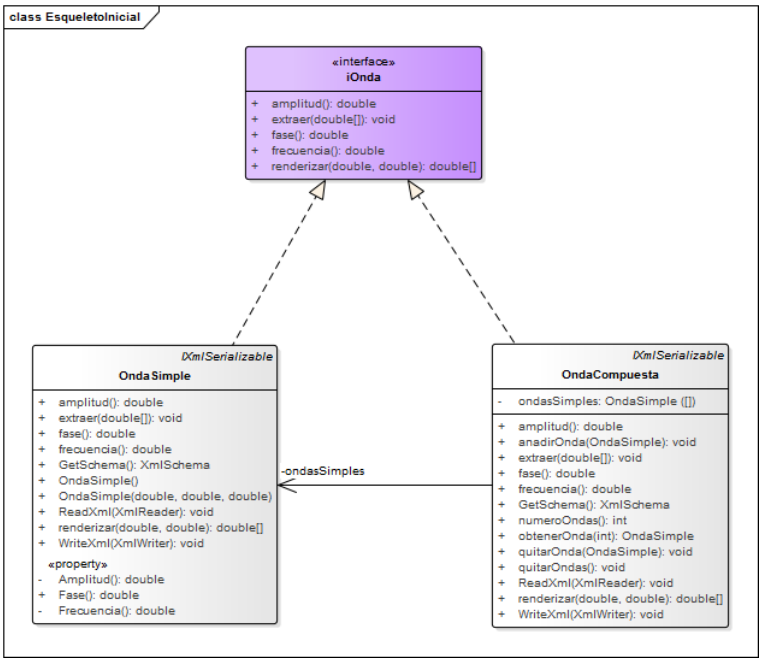


Figura 6.11: Diagrama de clases de gestión de ondas, tal como se implementaron en 'EsqueletoInicial'.

do un objeto ONDACOMPUESTA, que es capaz de extraer y crear los ONDASIMPLE que contenga, ya que por cada `<wave>` se genera una ONDASIMPLE asociada a la compuesta. Si se desea una onda simple, basta con incluir un solo elemento `<wave>`. A través de la función numeroOndas():int, se puede conocer de cuántas ondas simples está formada la compuesta y tratarla en consecuencia extrayendo el primer (y único, en ese caso) elemento del vector *ondasSimples* mediante la función obtenerOnda(0).

Una optimización introducida en este proyecto afecta a la forma en que las ondas se calculan. La idea es que, dado que estamos trabajando con formas de onda periódicas, sería conveniente, en vez de calcular un seno durante varios ciclos, resulta mucho más rápido calcular únicamente el primero, almacenar los valores en un vector temporal, y luego replicarle las veces necesarias hasta llegar al número de segundos o de ciclos que deba durar la onda. Esta afirmación puede demostrarse matemáticamente si tenemos en cuenta de nuevo la ecuación de las ondas sinusoidales:

$$y(t) = A \cdot sin(\omega t + \theta) \tag{6.8}$$

Calcular el seno anterior supone seis accesos a memoria, considerando también los registros. En primer lugar, supongamos que ω, θ y A ya están en registros. Es necesario obtener el valor de t y copiarle a un registro (1), multiplicarlo por omega y dejar el resultado en un registro (2), realizar la suma y dejarla en un registro (3), efectuar el seno(4), multiplicar el resultado por A (5), y moverlo a la memoria principal (6). Con la optimización, sólo sería necesario realizar esta operación durante el primer ciclo, ya que para el resto, únicamente hay que leer de una posición de memoria y escribir en otra. Además, ya que la mayoría de ordenadores modernos disponen de un módulo de DMA, ni tan siquera es necesario utilizar la CPU para realizar la copia, acelerando aún más el cálculo de la onda.

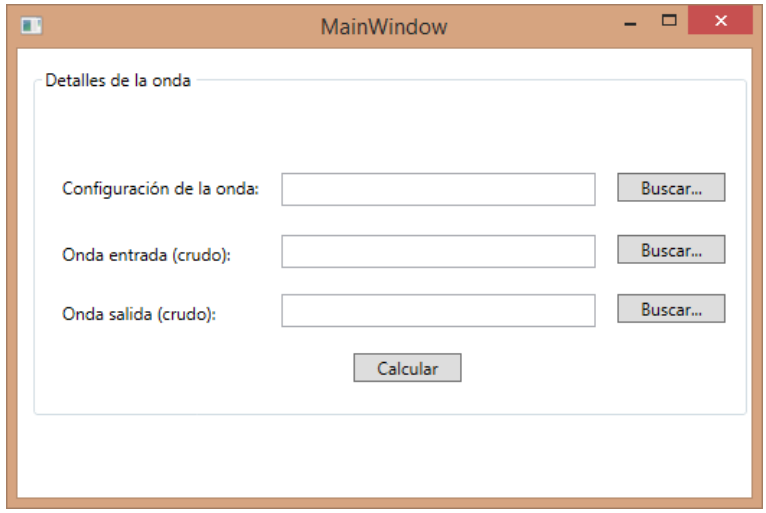


Figura 6.12: Aspecto del formulario principal y único del programa 'EsqueletoInicial'. Los textboxes segundo y tercero empezando por arriba no se usan.

Esto sólo funciona con señales sinusoidales simples y complejas, y no con barridos de frecuencia, evidentemente.

6.6. Cuarta iteración

La cuarta iteración corresponde a las mejoras realizadas sobre la tercera que se han descrito a lo largo de esta memoria. Por ejemplo, las pantallas diseñadas que se han mostrado y discutido en el apartado 5.4 pertenecen a este ciclo de desarrollo, aunque no están completamente implementadas en el código a fecha de redacción. También se ha realizado un diagrama de clases parcial que puede verse en la figura 6.13.

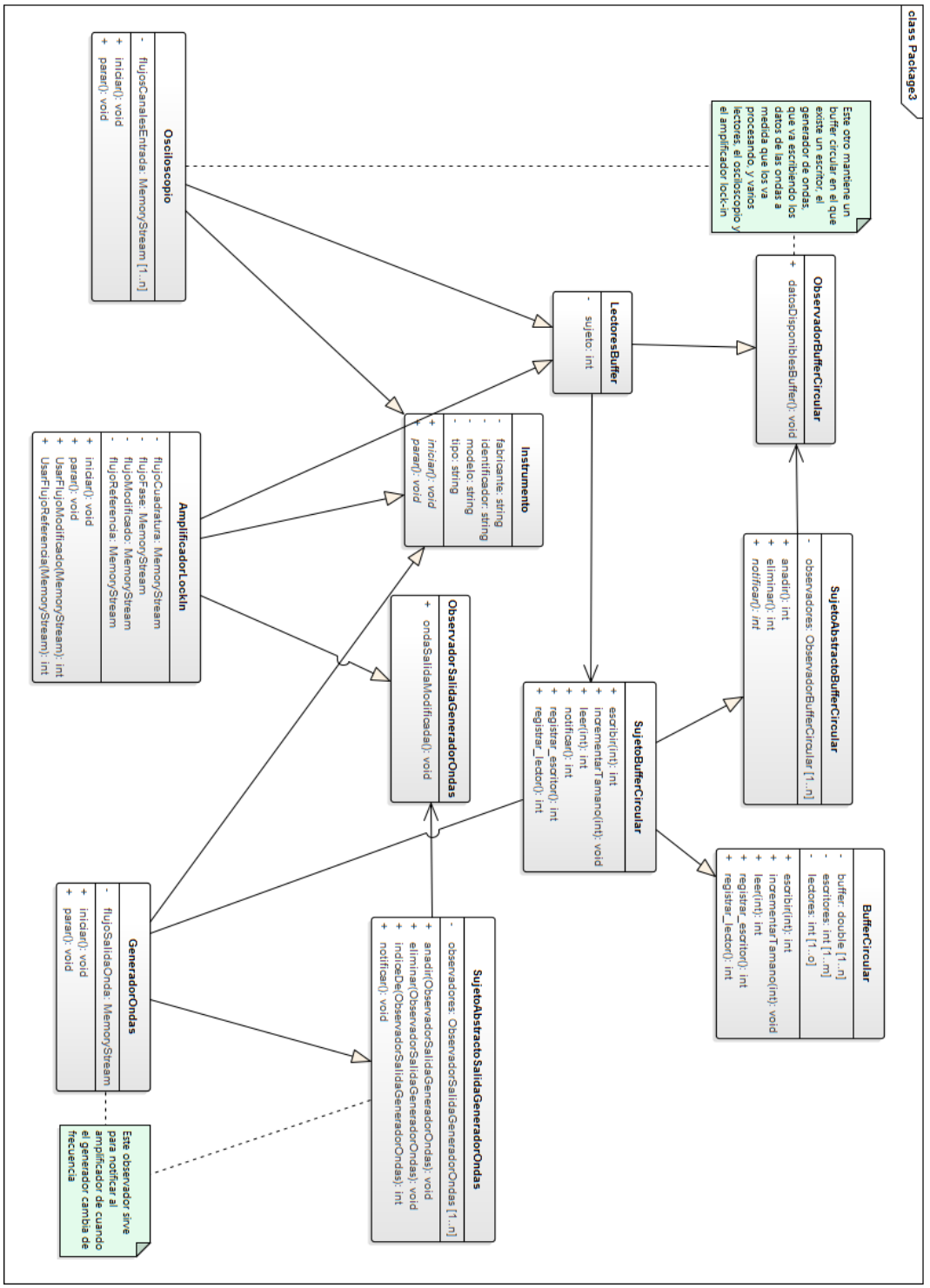


Figura 6.13: Diagrama de clases parcial de la cuarta iteración del desarrollo.

Capítulo 7
Conclusiones

"En alguna parte, algo increíble está esperando a ser descubierto."

Carl Sagan

Como cierre de la memoria, este capítulo pretende resumir el proyecto planteado, lo que se ha conseguido, y el trabajo pendiente. Aquí expondremos las dificultades a las que hemos tenido que hacer frente, el contexto en el que aparecieron y cómo fueron solucionadas. También se realiza una comparación entre los objetivos fijados al principio del proyecto y que fueron expuestos en los capítulos 1 y 4, los que finalmente se han implementado y en que proporción se ha realizado cada uno. A continuación se hace un breve resumen de lo que el proyecto ha supuesto para nosotros en términos de conocimento, para pasar, por último, a explicar los puntos en los que los objetivos no han podido llevarse a cabo y qué otras mejoras y extensiones pueden plantearse para extender la funcionalidad que hasta ahora tiene.

7.1. Dificultades encontradas

En esta sección se enumeran las dificultades a las que hemos tenido que hacer frente durante el desarrollo de este proyecto fin de máster, cómo y por qué surgieron, y de qué forma las solucionamos:

- Una vez más, nos hemos encontrado con bibliotecas externas relativamente complejas y con una documentación muy pobre. Esto nos ha sucedido utilizando los drivers IVI para tratar de controlar el osciloscopio Keysight. En un principio decidimos intentar desrrollar el proyecto con Matlab y la única referencia que conseguimos encontrar en la documentación que nos funcionara era un pequeño trozo de código sobre cómo detectar el osciloscopio y leer, de forma básica, información de un canal. Posteriormente logramos encontrar códigos de ejemplo para los osciloscopios de la gama InfiniiVision, lo que nos permitió avanzar pero a la hora de enviar órdenes al generador de ondas, las funciones se ejecutaban sin efecto alguno sobre él. No encontramos porqué sucedía esto, pero encontramos código de ejemplo en C# instalado con la Agilent IO Libraries Suite. Este fue el motivo para cambiarnos de lenguaje de programación.

- Durante el desarrollo con código en C#, siempre nos sucedía que la amplitud de la onda que salía del generador de funciones no coincidía con la que nosotros sintetizábamos, cosa que

verificábamos mediante el canal de referencia, que conectaba la salida del generador de funciones con la entrada 1 del osciloscopio. Nos pusimos en contacto con Keysight en España solicitándole soporte técnico sobre esta cuestión, pero nunca nos respondieron a los correos.

- Otro problema al que tuvimos que hacer frente fue la incompatibilidad de horarios. En algunos casos, como por ejemplo durante semana santa, resultaba difícil avanzar en el trabajo ya que la facultad se encuentra cerrada en esas fechas. Además, dado que éramos dos personas desarrollando código, resultaba muy tedioso tener que desarrollar trozos por separado para luego fusionarlos, sobre todo cuando formaban parte de la misma clase. Esto lo solucionamos montando el servidor Team Foundation Server en el laboratorio.

- Al principio del todo del proyecto, disponíamos en el laboratorio de un ordenador antiguo que apenas tenía potencia suficiente como para ejecutar Matlab, y se congelaba al abrir una segunda aplicación. Nuestros tutores solventaron el problema adquiriendo el ordenador i5 que está listado en en los costes.

- Aunque disponíamos del servidor de control de versiones, resultaba difícil seguir, ante un problema, qué estrategias habíamos intentado y porqué no habían dado resultado. Esto lo solucionamos cuando empezamos a utilizar el tablero de vinilo blanco del laboratorio, donde podíamos dejar los esquemas, diagramas de flujo y de clases, y algoritmos en pseudocódigo que se nos ocurrían, que ya habíamos probado, que estábamos probando, y los quedaban por verificar. Esto sucedió hacia mediados del segundo cuatrimestre.

- Dadas las experiencias que habíamos tenido con nuestra compañera Amalia en el HackForGood del 2014, se nos ocurrió que sería buena idea realizar descomposición de funcionalidad y establecer un tablero KanBan en la corchera. Resultó muy útil porque nos permitía llevar cuenta de cuánto faltaba del proyecto, y quién estaba haciendo qué tarea y detectar cuando uno de los dos estaba bloqueado en el desarrollo, de un sólo golpe de vista.

- Anteriormente a la realización del proyecto nos había sucedido que, cuando varias personas comparten información sobre soportes físicos, resultaba difícil tener control de quién tenía qué documentos. Por este motivo decidimos que todos los apuntes y esquemas que nos dieron los tutores debían ser guardados por una sola persona. Hacia finales del proyecto vimos que lo mejor era escanear la documentación y tenerla en el servidor GIT disponible, al igual que hacíamos con el código.

- Como se ha expuesto previamente otro de los problemas encontrados durante el desarrollo de la clase onda en la segunda iteración fue lograr que funcionara el detector de desfase a partir de un *array* de datos proveniente del renderizado del simulador de ondas, o del osciloscopio. Representando los vectores en papel y a través de varias trazas y pruebas, pudo descubrirse que el problema se encontraba en la función arcoseno.

- En varias ocasiones ha sido complicado encontrar y añadir los patrones y las estructuras de datos necesarias para que el sistema funcionara de la mejor forma posible, sobre todo con los diversos cambios en la prioridad de los requisitos tras cada iteración; llegando en algunos casos a tardar varias semanas en encontrar la solución. El punto bueno de esto es que eso ha provocado que el sistema esté diseñado de una forma completamente flexible, permitiendo añadir más instrumentos, por ejemplo.

- En general, este ha sido un proyecto duro debido a que nunca habíamos tenido ocasión de profundizar en el tema de procesamiento digital de señales. Afortunadamente hemos contado

con la colaboración de nuestros tutores, y de muchas otras personas, como Héctor García, Jesús Arias y Manuel González gracias a los cuales nos ha sido posible llevar el proyecto hasta su grado de desarrollo a escritura de estas líneas.

7.2. Objetivos alcanzados

En este apartado analizaremos la diferencia entre los objetivos marcados, los requisitos funcionales y funcionales encontrados durante la fase de análisis, y los casos de uso, y lo que finalmente ha sido implementado y testeado. De entre esas metas, algunas de las que se han cumplido han sido las siguientes:

- Se ha logrado extraer, a partir de dos señales sinusoidales (salida del sistema y referencia), las componentes en fase y cuadratura, y la capacitancia y conductancia del circuito bajo test a partir de ellas.
- Se ha podido diseñar un sistema orientado a objetos extensible, que permite la inclusión de múltiples tipos de dispositivo, entre los que ya forman parte el amplificador Lock-in, el generador de ondas, el osciloscopio y el registrador de datos. Además, el sistema entero se ha codificado en C#, que es una variante de C.
- Se ha conseguido implementar generador de señal que permite configurar, sintetizar y analizar frecuencias simples, complejas y barridos.
- Ha sido posible crear un conector en la aplicación de tal forma que se pudiera configurar y controlar un generador de ondas y un osciloscopio hardware; ambos dos incorporados en el osciloscopio Keysight DSO-X 3104A, para enviarle la configuración de una onda y recoger tanto la señal de referencia como la de salida del sistema.
- Se han creado y desarrollado algunas estrategias para optimizar los algoritmos, acelerando un poco su velocidad de ejecución.
- Hemos mejorado y aprendido mucho sobre teoría de la señal, procesamiento digital de señal, ruido y electrónica en general, extendiendo el conocimiento previamente adquirido en las asignaturas de Electrónica I y II, Sistemas digitales, Diseño de circuitos, Técnicas de mantenimiento de circuitos y Periféricos de la carrera. También hemos podido aplicar nuestros conocimientos en informática en el proceso. Discutiremos más sobre este aspecto en la próxima sección.

7.3. Conocimientos adquiridos

Es raro que un proyecto fin de carrera o trabajo fin de máster no aporte nada a las persona que los desarrollan. En este caso, el proyecto está ligado a muchos temas que son de nuestro interés, como son el audio digital, el procesado de imagen, la programación orientada a objetos, la planificación y gestión de proyectos, la electrónica, y los fundamentos y aplicaciones que hay detrás de ellos.

Esta es la primera vez que aprendemos de la existencia de los amplificadores Lock-in, un instrumento que tiene muchísima más importancia del que parece. Como ya hemos visto en la sección 2.3, tiene aplicación en química analítica, en física y en electrónica y telecomunicaciones. Durante

los cursos académicos anterior y actual he tenido la suerte de ver e incluso en algunos casos trabajar para documentar el manejo de instrumentación para análisis en química, como han podido ser espectrofotómetros UV-Vis y FTIR[1], este último incorporando una implementación de la transformada de Fourier como sus propias siglas indican.

También es la primera ocasión en la que trabajamos en un proyecto tan largo (de un año académico de duración) con otra persona, ya que ni en la carrera ni en el máster existen (¡por suerte!) asignaturas anuales. Ha resultado interesante ver la forma en la que dos personas podían ponerse de acuerdo, ya fuera compatibilizando horarios, escogiendo una metodología de trabajo, buscando bibliografía, asignando subtareas a cada uno, poniendo en común los diseños y realizando comentarios constructivos entre nosotros, que nos permitieran ver los fallos y elegir las mejores decisiones. También hemos podido aprender a montar un servidor con Team Foundation Server para desarrollo con Visual Studio y Bonobo Git Server para compartir documentación.

De la misma manera, nunca antes habíamos podido trabajar tan extensamente con instrumentación de laboratorio, como por ejemplo con el osciloscopio, ya que generalmente son instrumentos caros. En otras asignaturas de la ingeniería técnica, como por ejemplo en Sistemas Digitales y Periféricos, habíamos realizado un uso muy básico de los osciloscopios. Este proyecto nos ha servido para tener claros conceptos tan fundamentales como 'ancho de banda', y para refrescar otros que habíamos aprendido durante la carrera, sobre todo en Física y Electrónica I y II, como 'Impedancia', 'Diagrama de Fasores' y operaciones con corriente alterna. Es triste pero a medida que pasa el tiempo se nos olvidan conceptos que hemos aprendido por falta de uso, así que esta ha sido una oportunidad maravillosa para recordarlos.

Además de los conceptos de la carrera, hemos sido capaces de poner en práctica de una forma más extendida lo que hemos podido aprender en las asignaturas del Máster en Ingeniería Informática, del que forma parte este trabajo. Como se ha expuesto anteriormente, hemos utilizado técnicas del Diseño centrado en el usuario de Ingeniería de la Interacción, conceptos de metodologías Ágiles vistos en Gestión Económico-Financiera de Empresas y Proyectos de Base Tecnológica.

Es la segunda vez que empleo LaTeXpara redactar un documento tan largo, habiendo sido la primera ocasión el Trabajo Fin de Carrera. Generalmente utilizo este lenguaje para redactar los trabajos e informes de las asignaturas, además de otros textos para mí, pero no es habitual sobrepasar las cien páginas.

Vistos los problemas que nos han surgido durante el desarrollo, hemos sido de capaces de llevar a cabo bastantes objetivos de los marcados. Reconozco que en muchas ocasiones resulta desesperante pegarse con un problema durante semanas, desechando soluciones hasta dar con la mejor, incluso llegando a no verse capaz de continuar. Esto nos servirá, sin duda, para tener mejor resistencia en proyectos largos, algo imprescindible en el mundo laboral, en la que los *deadlines* son fijos y casi siempre excesivamente cortos.

Por último, hemos mejorado a la hora de buscar bibliografía especializada en electrónica. Hace muchos años que me estoy acostumbrado a leer artículos para conocer las novedades en el mundo de la ciencia, pero casi nunca han sido de electrónica. Los artículos tienen la peculiaridad que además

[1] FTIR es la abreviatura de *Fourier Transform InfraRed spectrophotometer*, espectrofotómetro infrarrojo con transformada de Fourier.

de leerlos, a veces también hay que saber interpretarlos correctamente.

7.4. Trabajo Futuro

Como suele suceder en los proyectos, a medida que va avanzando el desarrollo aparecen nuevas propuestas de mejora, subproyectos que se desestiman, y tareas pendientes con prioridad baja, que quedan apuntadas como trabajo futuro. En este caso, además, el trabajo no está totalmente terminada, sino que queda finalizar la migración y acabar la última iteración. Por tanto, las aportaciones necesarias o convenientes para el proyecto se resumen en los siguientes puntos:

- Debido a que se ha cortado el proyecto para evitar que se extendiera demasiado en el tiempo, es necesario terminar de desarrollar la aplicación, migrando el código de la tercera iteración a la cuarta, y escribiendo el código que falta. También sería necesario realizar un conjunto de pruebas unitarias suficiente como para verificar las funciones y las clases de la nueva iteración.
- Sería muy conveniente que, en vez de utilizar la barra de tareas del sistema operativo para buscar entre los instrumentos creados, se añadiera un elemento al menú principal del formulario principal de la aplicación. El problema de la barra de tareas reside en que los nombres de las ventanas son demasiado largos como para que los pueda ver el usuario, lo que obliga, en caso de que el número de instrumentos sea grande, a revisarlas una a una.

- Implementar nuevos tipos dispositivos, ampliando la lista de los ya existentes 'Amplificador Lock-in', 'Osciloscopio', 'Generador de ondas' y 'Registrador de datos'. El sistema está diseñado de tal forma que se puedan crear nuevas clases que hereden de IINSTRUMENTO e implementen sus funciones. También es necesario desarrollar los drivers para controlar, a través de GPIB, el amplificador Lock-in que estaba marcado en los requisitos y que no ha podido hacerse por falta de tiempo. Además de éste, sería interesante desarrollar las interfaces y los drivers de otros modelos de instrumentos.
- En el proyecto 'EsqueletoInicial' de la tercera iteración, sólo se ha utilizado la biblioteca NMath para el cálculo de los picos mediante el detector de Savitzky-Golay. Estas bibliotecas tienen la ventaja de apoyarse, a su vez, en las MKL (*Math Kernel Libraries*, de Intel, que están muy optimizadas para esa arquitectura, y con las que he trabajado en la asignatura de 'Infraestructuras para el desarrollo de aplicaciones de computación de altas prestaciones'. Sería conveniente buscar qué más puntos de la aplicación, sobre todo en el trabajo con vectores, pueden optimizarse utilizando otras funciones de esta biblioteca. Con el mismo objetivo, puede analizarse el rendimiento de los algoritmos y buscar otros más optimizados.
- A su vez, y en caso de que desee poder compilarlo para procesadores digitales de señal, conviene portar todo el código a C++ o C, implementando, dentro del proyecto, el código necesario de las funciones de las bibliotecas externas (como NMath) que se estén utilizando.
- Cabe la posibilidad de realizar una colaboración entre otros programas de simulación, por ejemplo en Linux, de tal forma que puedan extenderse mutuamente sus funcionalidades, a través de APIs.

- Por último, en caso de transformar el sistema software en algo más próximo al hardware mediante el DSP, sería muy interesante diseñar y fabricar un circuito impreso y conectarle una pantalla LCD o similar para controlarlo, creando un instrumento que combinara la funcionalidad de todos los cuatro que forman parte de la aplicación.

Bibliografía

[1] «2000, 3000, 4000, 6000 InfiniiVision X-Series Oscilloscope IVI and MATLAB Instrument Drivers | Keysight (Agilent)».
http://www.keysight.com/main/software.jspx?ckey=2019021&lc=spa&cc=ES&nid=-33573.1006712&id=2019021

[2] «33503A | BenchLink Waveform Builder Pro Software | Keysight Technologies».
http://es.rs-online.com/web/p/oscilloscope-software/7470891/

[3] «33503A BenchLink Waveform Builder Pro Software | Keysight (Agilent)».
http://www.keysight.com/en/pd-1962285-pn-33503A/benchlink-waveform-builder-pro-software?nid=-536902257.977229&cc=ES&lc=spa

[4] «Agilent Infiniium and InfiniiVision Oscilloscopes - MATLAB Example - File Exchange - MATLAB Central».
http://www.mathworks.com/matlabcentral/fileexchange/22485-agilent-infiniium-and-infiniivision-oscilloscopes---matlab-example

[5] «Algorithms (Peak Analyzer)».
http://www.originlab.com/doc/Origin-Help/PA-Algorithm

[6] «Amplitud (física)». Page Version ID: 82481808.
http://es.wikipedia.org/w/index.php?title=Amplitud_(física)&oldid=82481808

[7] «The Analog Lock-in Amplifier».
http://cpm.uncc.edu/sites/cpm.uncc.edu/files/media/tn1002.pdf

[8] «ARK | Intel Core i5-4460 Processor (6M Cache, up to 3.40 GHz)».
http://ark.intel.com/es-es/products/80817/Intel-Core-i5-4460-Processor-6M-Cache-u
40-GHz

[9] «asin(sin(3,91011620211879)) - Buscar con Google».
https://www.google.es/search?q=asin(sin(3,91011620211879))

[10] «Audio digital». Page Version ID: 82275778.
https://es.wikipedia.org/w/index.php?title=Audio_digital&oldid=82275778

[11] «Bonobo Git Server - Git Server for Windows».
https://bonobogitserver.com/

[12] «Dielectric spectroscopy». Page Version ID: 652971658.
https://en.wikipedia.org/w/index.php?title=Dielectric_spectroscopy&oldid=652971658

[13] «The Digital Lock-in Amplifier».
http://cpm.uncc.edu/sites/cpm.uncc.edu/files/media/tn1003.pdf

[14] «DSOX3104A Oscilloscope: 1 GHz, 4 Analog Channels | Keysight (Agilent)».
http://www.keysight.com/en/pd-2092275-pn-DSOX3104A/oscilloscope-1-ghz-4-analog-channels?cc=ES&lc=spa

[15] «Electrically detected magnetic resonance». Page Version ID: 549935511.
https://en.wikipedia.org/w/index.php?title=Electrically_detected_magnetic_resonance&oldid=549935511

[16] «Electron paramagnetic resonance». Page Version ID: 663196473.
https://en.wikipedia.org/w/index.php?title=Electron_paramagnetic_resonance&oldid=663196473

[17] «Enterprise Architect - Pricing and Purchasing».
http://www.sparxsystems.com.au/products/ea/purchase.html

[18] «Filtro paso bajo». Page Version ID: 78160694.
http://es.wikipedia.org/w/index.php?title=Filtro_paso_bajo&oldid=78160694

[19] «Filtro Sinc». Page Version ID: 76770958.
https://es.wikipedia.org/w/index.php?title=Filtro_Sinc&oldid=76770958

[20] «Finding Peaks in Data with NMath».
http://www.centerspace.net/blog/finding-peaks-in-data-with-nmath/

[21] «Identidades trigonométricas». Page Version ID: 83335046.
https://es.wikipedia.org/w/index.php?title=Identidades_trigonométricas&oldid=83335046

[22] «IO Libraries Suite | Keysight (Agilent)».
http://www.keysight.com/en/pd-1985909/io-libraries-suite?nid=-33330.977662.00&cc=ES&lc=spa&cmpid=zzfindiosuite

[23] «Lock-in amplifier». Page Version ID: 667575017.
https://en.wikipedia.org/w/index.php?title=Lock-in_amplifier&oldid=667575017

[24] «Low Light Signal Measurements - Physics 111-Lab Wiki».
http://labs.physics.berkeley.edu/mediawiki/index.php/Low_Light_Signal_Measurements

[25] «Math.Asin (Método) (System)».
https://msdn.microsoft.com/es-es/library/system.math.asin(v=vs.110).aspx

[26] «Math.Sin (Método) (System)».
https://msdn.microsoft.com/es-es/library/system.math.sin(v=vs.110).aspx

[27] «Mezclador de frecuencias». Page Version ID: 83225398.
https://es.wikipedia.org/w/index.php?title=Mezclador_de_frecuencias&oldid=83225398

[28] «Microsoft Office 365: More new packages and prices coming in November».
http://www.zdnet.com/article/microsoft-office-365-more-new-packages-and-prices-co

[29] «Model 5210 Dual Phase Lock-in Amplifier - Instruction Manual».
http://www.princetonappliedresearch.com/download/5210-5210EC.pdf

[30] «Nuclear Magnetic Resonance - Physics 111-Lab Wiki».
http://www.advancedlab.org/mediawiki/index.php/Nuclear_Magnetic_Resonance

[31] «Problema al cargar la página».
http://doi.wiley.com.sci-hub.bz/10.1002/asia.201500560

[32] «RC time constant». Page Version ID: 646605329.
http://en.wikipedia.org/w/index.php?title=RC_time_constant&oldid=646605329

[33] «Resonancia paramagnética electrónica». Page Version ID: 74268504.
https://es.wikipedia.org/w/index.php?title=Resonancia_paramagnética_electrónica&oldid=74268504

[34] «Ruido rosa». Page Version ID: 71832206.
https://es.wikipedia.org/w/index.php?title=Ruido_rosa&oldid=71832206

[35] «Savitzky-Golay Smoothing in C#».
http://www.centerspace.net/blog/savitzky-golay-smoothing/

[36] «Servicio de Contratación » Expediente nº 26/11».
http://contratacion.umh.es/2012/02/22/expediente-no-2611/

[37] «Shared Components».
http://www.ivifoundation.org/shared_components/Default.aspx

[38] «Specifying Lock-in Amplifiers».
http://cpm.uncc.edu/sites/cpm.uncc.edu/files/media/tn1001.pdf

[39] «Unit circle». Page Version ID: 665943635.
http://en.wikipedia.org/w/index.php?title=Unit_circle&oldid=665943635

[40] «USB Monitor Pro».
http://www.fabulatech.com/usb-monitor-pro-purchase.html

[41] «Visual Studio Ultimate with MSDN 2012».
http://www.amazon.com/Visual-Studio-Ultimate-MSDN-2012/dp/B008RVZOIC

[42] «VStudio Foundation Server 2012».

[43] «What is a Lock-in Amplifier?».
http://cpm.uncc.edu/sites/cpm.uncc.edu/files/media/tn1000.pdf

[44] ABU_AL_AISH, A.; REHMAN, M.; ABDULLAH, M. y ABU HASSAN, A.:. «Microcontroller Based Capacitive Mass Measuring System». **10(1)**, pp. 15–18. doi: 10.2478/v10048-010-0003-9.
http://www.degruyter.com/view/j/msr.2010.10.issue-1/v10048-010-0003-9/v10048-010-0003-9.xml

[45] BEASLEY, JEFFREY S.; HYMER, JONATHAN D. y MILLER, GARY M.:. Electronic Communications: A System Approach. Prentice Hall, 1ª edición. ISBN 9780132988636.

[46] BEASLEY, JEFFREY S. y MILLER, GARY M.:. Modern Electronic Communication. Pearson Education Limited, pearson new international edition edition[a] edición. ISBN 9781292025476.

[47] BINNIG, G.; QUATE, C. F. y GERBER, CH.:. «Atomic Force Microscope». **56(9)**, pp. 930–933. doi: 10.1103/PhysRevLett.56.930.
http://link.aps.org/doi/10.1103/PhysRevLett.56.930

[48] BLAIR, D. P. y SYDENHAM, P. H.:. «Phase sensitive detection as a means to recover signals buried in noise». **8(8)**, p. 621. ISSN 0022-3735. doi: 10.1088/0022-3735/8/8/001.
http://iopscience.iop.org/0022-3735/8/8/001

[49] BOLETÍN OFICIAL DEL ESTADO:. «Anuncio 26318 del BOE núm. 178 de 2012 - BOE-B-2012-26318».
http://www.boe.es/boe/dias/2012/07/26/pdfs/BOE-B-2012-26318.pdf

[50] BOYLESTAD, ROBERT L. y NASHELSKY, LOUIS:. Electrónica: Teoría de circuitos y dispositivos electrónicos. Pearson Educación, 10ª edición. ISBN 9786074422924.

[51] BUD, ROBERT y WARNER, DEBORAH JEAN:. Instruments of Science: An Historical Encyclopedia. Taylor & Francis. ISBN 9780815315612.

[52] BURDETT, RICHARD:. «Amplitude Modulated Signals: The Lock-in Amplifier». En: Handbook of Measuring System Design, pp. 1198 – 1208. John Wiley & Sons, 1ª edición. ISBN 9780470021439.

[53] CLARKSON, P.; ESWARD, T. J.; HARRIS, P. M.; SMITH, A. A. y SMITH, I. M.:. «Software simulation of a lock-in amplifier with application to the evaluation of uncertainties in real measuring systems». **21(4)**, p. 045106. ISSN 0957-0233. doi: 10.1088/0957-0233/21/4/045106.
http://iopscience.iop.org/0957-0233/21/4/045106

[54] COSENS, C. R.:. «A balance-detector for alternating-current bridges». **46(6)**, p. 818. ISSN 0959-5309. doi: 10.1088/0959-5309/46/6/310.
http://iopscience.iop.org/0959-5309/46/6/310

[55] COVA, S.; LONGONI, A. y FREITAS, I.:. «Versatile digital lock?in detection technique: Application to spectrofluorometry and other fields». **50(3)**, pp. 296–301. ISSN 0034-6748, 1089-7623. doi: 10.1063/1.1135819.
http://scitation.aip.org/content/aip/journal/rsi/50/3/10.1063/1.1135819

[56] DE MARCELLIS, A.; DI GIANSANTE, A.; FERRI, G.; DI NATALE, C.; MARTINELLI, E. y D'AMICO, A.:. «Analog automatic lock-in amplifier for very low gas concentration detection». **5**, pp. 200–203. ISSN 1877-7058. doi: 10.1016/j.proeng.2010.09.082.
http://www.sciencedirect.com/science/article/pii/S1877705810006296

[57] DENING, D. C:. «Simulation of a digital lock-in amplifier». **23(3)**, pp. 461–463. ISSN 0022-2364. doi: 10.1016/0022-2364(76)90279-1.
http://www.sciencedirect.com/science/article/pii/0022236476902791

[58] DODGE, CHARLES y JERSE, THOMAS A.:. Computer Music: Synthesis, Composition, and Performance. Schirmer, 2 editionª edición. ISBN 9780028646824.

[59] DOMÉNECH, ÁLVARO:. «Efectos de sonido: la ecualización».
http://www.disca.upv.es/adomenec/IASPA/tema4/ES-Filtros.html

[60] GABAL, M.; MEDRANO, N.; CALVO, B.; MARTÍNEZ, P. A.; CELMA, S. y VALERO, M. R.:. «A complete low voltage analog lock-in amplifier to recover sensor signals buried in noise for embedded applications». **5**, pp. 74–77. ISSN 1877-7058. doi: 10.1016/j.proeng.2010.09.051.
http://www.sciencedirect.com/science/article/pii/S1877705810005989

[61] GASPAR, JAVIER; CHEN, SUEI FENG; GORDILLO, ALEJANDRO; HEPP, MATEO; FERREYRA, PABLO y MARQUÉS, CARLOS:. «Digital lock in amplifier: study, design and development with a digital signal processor». **28(4)**, pp. 157–162. ISSN 0141-9331. doi: 10.1016/j.micpro.2003.12.002.
http://www.sciencedirect.com/science/article/pii/S0141933104000079

[62] GEIST, JON:. «Waveform-Independent Lock-In Detection». **43(11)**, pp. 1704–1705. ISSN 0034-6748, 1089-7623. doi: 10.1063/1.1685531.
http://scitation.aip.org/content/aip/journal/rsi/43/11/10.1063/1.1685531

[63] GIGLER, ALEXANDER M.; DIETZ, CHRISTIAN; BAUMANN, MAXIMILIAN; MARTINEZ, NICOLÁS F.; GARCÍA, RICARDO y STARK, ROBERT W.:. «Repulsive bimodal atomic force microscopy on polymers». **3(1)**, pp. 456–463. ISSN 2190-4286. doi: 10.3762/bjnano.3.52.
http://www.beilstein-journals.org/bjnano/content/3/1/52/abstract

[64] GREENE, CHERYL y MEIDT, SHARON E.:. «Photoluminescence Excitation Spectroscopy of Semiconductor Quantum Wells».
http://www.phy.davidson.edu/StuHome/shmeidt/JuniorLab/QuantumWells/QuantumWells.htm

[65] GUTIÉRREZ, MIGUEL AGUILAR:. Bioelectromagnetismo: campos eléctricos y magnéticos y seres vivos. Editorial CSIC - CSIC Press. ISBN 9788400079284.

[66] HOEHNE, FELIX; DREHER, LUKAS; BEHRENDS, JAN; FEHR, MATTHIAS; HUEBL, HANS; LIPS, KLAUS; SCHNEGG, ALEXANDER; SUCKERT, MAX; STUTZMANN, MARTIN y BRANDT, MARTIN S.:. «Lock-in detection for pulsed electrically detected magnetic resonance». **83(4)**, p. 043907. ISSN 0034-6748, 1089-7623. doi: 10.1063/1.4704837.
http://scitation.aip.org/content/aip/journal/rsi/83/4/10.1063/1.4704837

[67] HUSSAIN, ZAHIR M.; SADIK, AMIN Z. y O'SHEA, PETER:. Digital Signal Processing: An Introduction with MATLAB and Applications. Springer, 2011 editionª edición. ISBN 9783642155901.

[68] JIMÉNEZ CALVO, JOSÉ ANTONIO:. «Ruido intrínseco en dispositivos electrónicos».
http://www.depeca.uah.es/depeca/repositorio/asignaturas/32421/Ruido.pdf

[69] JURE, LUIS; LÓPEZ, ERNESTO y ROCAMORA, MARTÍN:. «Clase 10: Introducción a los Filtros Digitales».
http://www.eumus.edu.uy/eme/ensenanza/electivas/dsp/presentaciones/clase10.pdf

[70] KAN, S.; GONORD, P.; FAN, M.; SAUZADE, M. y COURTIEU, J.:. «Automatic NMR field-frequency lock-pulsed phase locked loop approach». **49(6)**, pp. 785–789. ISSN 0034-6748, 1089-7623. doi: 10.1063/1.1135615.
http://scitation.aip.org/content/aip/journal/rsi/49/6/10.1063/1.1135615

[71] KOLLE, C. y O'LEARY, P.:. «Low-cost, high-precision measurement system for capacitive sensors». **9(3)**, p. 510. ISSN 0957-0233. doi: 10.1088/0957-0233/9/3/028.
http://iopscience.iop.org/0957-0233/9/3/028

[72] LADEFOGED, PETER:. Elements of Acoustic Phonetics. Univ of Chicago Pr, edición: revised.ª edición. ISBN 9780226467641.

[73] LEIS, J.; KELLY, C. y BUTTSWORTH, D.:. «Sampling, quantization and computational aspects of the quadrature lock-in amplifier». En: 2012 6th International Conference on Signal Processing and Communication Systems (ICSPCS), pp. 1–7. doi: 10.1109/ICSPCS.2012.6507974.

[74] LEIS, J.; MARTIN, P. y BUTTSWORTH, D.:. «Simplified digital lock-in amplifier algorithm». **48(5)**, pp. 259–261. ISSN 0013-5194. doi: 10.1049/el.2012.0193.
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6164313

[75] LEÓN, MARIO:. Diccionario de informática, telecomunicaciones y ciencias afines: Inglés-Español / Español-Inglés. Ediciones Díaz de Santos, S.A., 1ª edición. ISBN 9788479786267.

[76] LIU, SHAOYANG y WANG, YIFEN:. «Chapter 6 - A Review of the Application of Atomic Force Microscopy (AFM) in Food Science and Technology». En: Steve L. Taylor (Ed.), Advances in Food and Nutrition Research, volumen 62, pp. 201–240. Academic Press.
http://www.sciencedirect.com/science/article/pii/B9780123859891000065

[77] LUO, JIANWEN; YING, KUI; HE, PING y BAI, JING:. «Properties of Savitzky-Golay digital differentiators». **15(2)**, pp. 122–136. ISSN 1051-2004. doi: 10.1016/j.dsp.2004.09.008.
http://www.sciencedirect.com/science/article/pii/S1051200404000727

[78] MARTÍN FERNÁNDEZ, MARCOS:. «Señales aleatorias y ruido».
http://lmi.bwh.harvard.edu/papers/pdfs/1996/martin-fernandezCOURSE96b.pdf

[79] MEADE, M. L.:. Lock-In Amplifiers: Principles and Applications. Inspec/Iee. ISBN 9780906048948.

[80] MOSKAU, DETLEF:. «Application of real time digital filters in NMR spectroscopy». **15(2)**, pp. 164–176. ISSN 1099-0534. doi: 10.1002/cmr.10031.
http://onlinelibrary.wiley.com/doi/10.1002/cmr.10031/abstract

[81] PROBST, P. A. y COLLET, B.:. «Low-frequency digital lock-in amplifier». **56(3)**, pp. 466–470. ISSN 0034-6748, 1089-7623. doi: 10.1063/1.1138324.
http://scitation.aip.org/content/aip/journal/rsi/56/3/10.1063/1.1138324

[82] PROJECT MANAGEMENT INSTITUTE:. Guía de los fundamentos para la dirección de proyectos / Guide to the Fundamentals of Project Management. Project Management Inst, 5 editionª edición. ISBN 9781628250091.

[83] SCHOLKMANN, FELIX; BOSS, JENS y WOLF, MARTIN:. «An Efficient Algorithm for Automatic Peak Detection in Noisy Periodic and Quasi-Periodic Signals». **5(4)**, pp. 588–603. doi: 10.3390/a5040588.
http://www.mdpi.com/1999-4893/5/4/588

[84] SCOFIELD, JOHN H.:. «Frequency?domain description of a lock?in amplifier». **62(2)**, pp. 129–133. ISSN 0002-9505, 1943-2909. doi: 10.1119/1.17629.
http://scitation.aip.org/content/aapt/journal/ajp/62/2/10.1119/1.17629

[85] SMITH, STEVEN:. Digital Signal Processing: A Practical Guide for Engineers and Scientists. Newnes, 1 editionª edición. ISBN 9780750674447.

[86] SONNAILLON, MAXIMILIANO OSVALDO y BONETTO, FABIÁN JOSE:. «A low-cost, high-performance, digital signal processor-based lock-in amplifier capable of measuring multiple frequency sweeps simultaneously». **76(2)**, p. 024703. ISSN 0034-6748, 1089-7623. doi: 10.1063/1.1854196.
http://scitation.aip.org/content/aip/journal/rsi/76/2/10.1063/1.1854196

[87] SONNAILLON, M.O.; URTEAGA, R. y BONETTO, FABIAN J.:. «High-Frequency Digital Lock-In Amplifier Using Random Sampling». **57(3)**, pp. 616–621. ISSN 0018-9456. doi: 10.1109/TIM.2007.911584.

[88] STANFORD RESEARCH SYSTEMS, INC:. «Model SR850 DSP Lock-In Amplifier».
http://www.thinksrs.com/downloads/PDFs/Manuals/SR850m.pdf

[89] STONE, ZACHARY:. «Building a Lock-In Based Pulse NMR Spectrometer».
https://web2.ph.utexas.edu/~markweb/PRESENTATION/Building_a_Lock_In_Based_Pulse_NMR_Spectrometer_Zack_Stone.pdf

[90] STORR, WAYNE:. «Low Pass Filter - Passive RC Filter Tutorial».
http://www.electronics-tutorials.ws/filter/filter_2.html

[91] SYDENHAM, PETER H. y THORN, R.:. Handbook of Measuring System Design. John Wiley & Sons, 1ª edición. ISBN 9780470021439.

[92] T., TORBJØRN:. «tikz pgf - unit circle sin(x) radians on x-axe - TeX - LaTeX Stack Exchange».
http://tex.stackexchange.com/questions/152054/unit-circle-sinx-radians-on-x-axe

[93] TEMPLE, PAUL A.:. «An introduction to phase-sensitive amplifiers: An inexpensive student instrument». **43(9)**, pp. 801–807. ISSN 0002-9505, 1943-2909. doi: 10.1119/1.9690.
http://scitation.aip.org/content/aapt/journal/ajp/43/9/10.1119/1.9690

[94] THIERET, T. E; MCNALLY, J; KRAFT, K; SONNENFROH, D y KREILICK, R. W:. «Correction of lineshape distortions from pulsed NMR experiments with lock-in amplifier or boxcar integrator detection». **39(2)**, pp. 203–206. ISSN 0022-2364. doi: 10.1016/0022-2364(80)90129-8.
http://www.sciencedirect.com/science/article/pii/0022236480901298

[95] TOMASI, WAYNE:. Sistemas de comunicaciones electrónicas. Pearson Educación. ISBN 9789702603160.

[96] UNIVERSIDAD DE VALLADOLID:. «Calendario Académico 2014/2015». http://www.uva.es/export/sites/uva/7.comunidaduniversitaria/7.06.calendarioacademico/_documentos/Calendario-14-15.pdf

[97] VENTZAS, DIMITRIOS y PETRELLIS, NIKOLAOS:. «Peak Searching Algorithms and Applications». doi: 10.2316/P.2011.738-049. http://www.actapress.com/Abstract.aspx?paperId=452138

[98] WANG, XIAOYI:. «Sensitive digital lock?in amplifier using a personal computer». **61(7)**, pp. 1999–2001. ISSN 0034-6748, 1089-7623. doi: 10.1063/1.1141413. http://scitation.aip.org/content/aip/journal/rsi/61/7/10.1063/1.1141413

[99] WEISSTEIN, ERIC W.:. «Inverse Sine». http://mathworld.wolfram.com/InverseSine.html

[100] WOLFSON, RICHARD:. «The lock?in amplifier: A student experiment». **59(6)**, pp. 569–572. ISSN 0002-9505, 1943-2909. doi: 10.1119/1.16824. http://scitation.aip.org/content/aapt/journal/ajp/59/6/10.1119/1.16824

[101] WOLLAN, DAVID S.:. «Comments on Lock?In Detection with NMR Saturation in Solids». **42(5)**, pp. 682–683. ISSN 0034-6748, 1089-7623. doi: 10.1063/1.1685200. http://scitation.aip.org/content/aip/journal/rsi/42/5/10.1063/1.1685200

[102] YANG, WUQIANG:. «Teaching phase-sensitive demodulation for signal conditioning to undergraduate students». **78(9)**, pp. 909–915. ISSN 0002-9505, 1943-2909. doi: 10.1119/1.3428642. http://scitation.aip.org/content/aapt/journal/ajp/78/9/10.1119/1.3428642