
Máster en Ingeniería de Sistemas y Control

# Amplificador lock-in basado en un microcontrolador

Alumno: D. Ignacio Horcas Calvo

Directores: Dr. José Sánchez Moreno
Dr. David Moreno Salinas

Curso 2021/22. Convocatoria de septiembre


Máster en Ingeniería de Sistemas y Control

# Amplificador lock-in basado en un microcontrolador

Tipo B: Proyecto específico propuesto por el alumno

Alumno: D. Ignacio Horcas Calvo

Directores: Dr. José Sánchez Moreno
Dr. David Moreno Salinas

A rellenar por el tribunal calificador

# Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado:

Firmado por HORCAS CALVO IGNACIO - ***7476** el día 04/09/2022 con un certificado emitido por AC FNMT Usuarios

## RESUMEN DEL PROYECTO

El proyecto consiste en el desarrollo de un prototipo de un amplificador lock-in basado en un microcontrolador, a diferencia de los basados en FPGA, que son la tendencia actual. Los amplificadores lock-in son instrumentos ampliamente empleados en laboratorios de física e ingeniería y su invención se remonta a la década de 1940. Con la evolución de la electrónica las versiones analógicas iniciales han dado paso a otras digitales, siendo ahora así prácticamente la totalidad de los modelos comerciales. En este trabajo aprovechamos los últimos desarrollos en el campo de los microcontroladores, presentando un prototipo funcional con una potencia de cálculo similar a otros amplificadores comerciales basados en otras tecnologías de mayor coste y complejidad.

Los amplificadores lock-in han ido evolucionando desde su invención, añadiendo opciones antes no existentes para adaptarse a nuevos usos, mejorando su eficiencia, la calidad de la parte analógica, etc. En el trabajo presentamos simulaciones de varias configuraciones, de mayor o menor complejidad, antes de pasar al diseño, fabricación y validación de una solución concreta que emplea un microcontrolador de doble núcleo como procesador digital de señales. Entre las ventajas que ofrece esta implementación destacan su bajo coste, la posibilidad de adaptar las funcionalidades a distintas aplicaciones o usos y en general la facilidad de desarrollo que ofrecen los sistemas de desarrollo integrados actuales, permitiendo programar en lenguajes de alto nivel e incluso depurar los programas estableciendo puntos de parada, visualizar desde el ordenador los valores de las variables de una manera cómoda, ejecutar las instrucciones de manera individual o por bloques, etc. En cuanto al *hardware*, es habitual que los microcontroladores modernos incluyan dos núcleos de alta eficiencia, unidades aritmético-lógicas con soporte para operaciones en coma flotante, gran variedad de periféricos para controlar distintos puertos en segundo plano sin usar tiempo de CPU, acceso directo a memoria, etc. Hemos aprovechado algunas de estas características para mejorar la eficiencia del código y conseguir altas frecuencias de muestreo y consecuentemente de las señales a analizar.

Para mejorar la calidad de las señales analógicas se han empleado convertidores externos, de manera que la parte analógica y la digital puedan estar mejor aisladas entre ellas que si compartieran la misma pastilla. Algunas de estas señales analógicas permiten comunicar el prototipo con otros aparatos de una manera sencilla, mientras que la configuración e incluso una lectura de datos a baja frecuencia se realizan digitalmente a través de un puerto USB.

**PALABRAS CLAVE**

Amplificador lock-in, microcontrolador, sistemas empotrados, *system on chip*, instrumentación científica, adquisición de datos, procesado de señales, amplitud, fase, oscilación.

**ÍNDICE**

## LISTA DE FIGURAS

## LISTA DE TABLAS

# CAPÍTULO 1. INTRODUCCIÓN

## 1.1 Contexto del proyecto

Este Trabajo Fin de Máster (TFM) ha sido realizado por el alumno D. Ignacio Horcas Calvo para optar al título de Máster en Ingeniería de Sistemas y Control, por acuerdo con los directores del proyecto, los doctores D. José Sánchez Moreno y D. David Moreno Salinas, profesores pertenecientes al departamento de Informática y Automática de la Universidad Nacional de Educación a Distancia (UNED).

En los siguientes apartados expondremos los motivos que nos han llevado a realizar este proyecto, así como el contexto en el que se ha desarrollado, desde dos puntos de vista distintos. Por una parte, presentaremos el contexto académico que acabamos de adelantar, pero también expondremos el contexto tecnológico, que ha sido de gran importancia para la mayoría de las decisiones que se han tomado durante el desarrollo del trabajo.

### 1.1.1 Contexto académico

Como hemos adelantado, el presente trabajo constituye el TFM del Máster en Ingeniería de Sistemas y Control, una titulación que se imparte de manera conjunta entre la Universidad Complutense de Madrid (UCM) y la UNED, en la que el alumno puede elegir entre asignaturas muy variadas con un punto de unión que es la teoría de control. Cada una de estas asignaturas puede tener un enfoque más teórico o práctico y trata temas muy concretos. Sin ninguna duda, una de las que más relación tiene con el trabajo que presentamos es “Sistemas Empotrados”, cuyo equipo docente lo forman los directores de este TFM, aunque también tiene gran carga de “Procesado de Señales” y de “Simulación de sistemas”. Si no entramos en los detalles concretos de lo que es un amplificador lock-in, podemos decir que hemos simulado y desarrollado un sistema empotrado para procesar señales. Hemos querido poner en práctica gran parte de los conocimientos adquiridos en estas asignaturas, pero no limitarnos a ellos. Aplicaremos técnicas de control híbrido, ya que estamos ante un sistema muestreado que se comunica con el mundo real, continuo por naturaleza, y para la correcta simulación debemos modelar previamente el propio amplificador y el sistema físico que emplearemos para validarlo.

### 1.1.2 Contexto tecnológico

En un entorno en el que la tecnología y las telecomunicaciones se han convertido en una parte fundamental de nuestra vida cotidiana, la electrónica es la base de muchos dispositivos que usamos a diario. Hoy en día, términos como domótica forman parte de nuestro lenguaje, nutrido por otros muchos relativos a la tecnología, la mayoría de ellos anglicismos adaptados o no al español. Así es común tener un teléfono móvil o una tableta (*tablet*) conectados a internet las 24 horas del día, que lo mismo sirven para sustituir al periódico que para guiarnos dándonos indicaciones por caminos desconocidos mientras conducimos. Los accesorios tecnológicos inundan nuestra vida,

ayudándonos a controlar nuestro ritmo cardíaco mientras hacemos deporte o animándonos a levantarnos si llevamos demasiado tiempo en actividades sedentarias. La miniaturización y reducción de costes y del consumo energético ha conducido a que hoy llevemos en el bolsillo o en la muñeca máquinas más potentes que aquellos primeros ordenadores que ocupaban habitaciones enteras y consumían cantidades de energía inconcebibles actualmente, aquellos en los que un *bug* podía ser realmente un bicho en una válvula y no un fallo del programador.

Los avances de las últimas décadas en el campo de los microprocesadores y microcontroladores han sido espectaculares, consiguiendo que esta tecnología sea asequible y accesible al gran público. Plataformas como Arduino [1] o Raspberry Pi [2] han popularizado el uso de dispositivos baratos y versátiles para funciones muy variadas. La sencillez de uso, la potencia y el precio ajustado han llevado a que, con pocos conocimientos previos, sin herramientas caras de programación de microcontroladores y menos de 20€, podamos conectar una de estas placas a nuestro ordenador con un cable USB (*Universal Serial Bus*), seguir un tutorial de internet y desarrollar nuestro propio *firmware*, aunque sólo sea cambiar la cadencia de parpadeo de un LED (*Light Emitting Diode*). Hace no demasiados años era impensable que estudiantes de instituto sin especial interés en la tecnología fueran capaces de programar un microcontrolador y hoy a muchos de ellos se les exige como parte de su formación. A esta expansión han contribuido repositorios de *software* gratuitos, como por ejemplo GitHub [3], propiciando que muchos desarrolladores a lo largo y ancho del planeta compartan su trabajo. Tanto particulares como empresas dejan disponibles distintos proyectos, bibliotecas y utilidades en general que facilitan el uso de componentes o módulos electrónicos. Las siglas anglosajonas DIY (*Do It Yourself*) han saltado del bricolaje tradicional a lo que podríamos llamar el bricolaje electrónico.

El campo de la instrumentación científica no se libra de esta tendencia a lo digital. Los aparatos puramente analógicos han ido dejando paso de manera progresiva a otros en los que el procesamiento de la señal lo realiza un DSP (*Digital Signal Processor*), ya sea en forma de FPGA (*Field Programmable Gate Array*), microcontrolador o el clásico procesador que incluye instrucciones máquina específicas para ciertas operaciones, algunas tan exóticas como calcular de manera aproximada el inverso de la raíz cuadrada de un número**.** En general es más sencillo y versátil disponer de un buen convertidor de analógico a digital (ADC, *Analog to Digital Converter*) procesar digitalmente y emplear un convertidor de digital a analógico (DAC, *Digital to Analog Converter*) para volver al mundo real que realizar complejos circuitos analógicos. Si este procesador digital es configurable, como los indicados, las actualizaciones son muy sencillas, permitiendo corregir errores, implementar mejoras o añadir nueva funcionalidad con solo cambiar el programa que se ejecuta en el sistema empotrado, lo que se conoce como el *firmware*.

Nos gustaría no tener que nombrar otra de las situaciones tecnológicas actuales, que esperemos se solucione pronto, y es la escasez de componentes electrónicos a nivel mundial. Desde hace varios años y de manera más aguda en los últimos meses, las fábricas no son capaces de satisfacer la demanda, el transporte cada vez es más caro y en general para estos componentes en un único sentido, desde Asia hacia el resto del

mundo, y cada poco tiempo oímos en las noticias que alguna fábrica, por ejemplo, de automóviles, ha tenido que parar su producción o reducir turnos porque no tienen suficientes microchips para montar. Intentando evitar este problema en la medida de lo posible se han priorizado componentes con un amplio stock en tiendas de distribución habituales [4]–[6]. Aun así, hemos tenido que diseñar en base al *stock* más que en base al catálogo de componentes de los distintos fabricantes, algo que cada vez es más común.

## 1.2 Motivación y objetivos

El amplificador lock-in se inventó en la primera mitad del siglo pasado. Aunque tradicionalmente se ha considerado que lo inventó el físico Robert H. Dicke en la década de 1940, que fue quien se encargó de su desarrollo y comercialización, en una entrevista de 1985 comentó haber leído algún artículo anterior donde empleaban esta técnica [7]. La primera idea en esta línea se publicó en 1934 [8] y se desarrolló con más detalle en 1941 [9]. Sea como fuere, la tecnología ha evolucionado mucho desde entonces, y los amplificadores se han ido adaptando a las nuevas tecnologías. Las versiones puramente analógicas iniciales, con muchas etapas de acondicionamiento, filtrado y amplificación, han dejado paso a otras donde todos los cálculos se realizan digitalmente y la parte analógica se limita algunas amplificaciones y filtros muy sencillos.

Nuestro objetivo es desarrollar un amplificador que, siendo básico, sea útil para aplicaciones que no necesiten frecuencias muy altas, que son la mayoría de ellas, y en las que la relación señal/ruido (*SNR, Signal to Noise Ratio*) no sea crítica. El precio de un amplificador comercial así suele ser de varios miles de euros. Incluso queremos ir más lejos y desarrollar una plataforma que permita implementar de manera sencilla sistemas de adquisición de datos y control con bajos tiempos de respuesta.

Antes del auge de las FPGA se solían emplear DSP secuenciales que controlaban ADC y DAC. El empleo de lenguajes de alto nivel, normalmente C, facilitaba el desarrollo y reducía el tiempo y los costes asociados a esta fase. La Figura 1 [10] muestra un esquema de distintas opciones de procesadores digitales ordenados según su rendimiento y flexibilidad. En general podemos considerar que el tiempo de desarrollo tiene una tendencia opuesta a la flexibilidad, en este caso aumentaría de derecha a izquierda. Los microcontroladores actuales tienen potencias de cálculo asombrosas, que creemos que pueden sustituir a estos DSP en muchas funciones, por lo que, unido a la mayor flexibilidad y facilidad de uso, han sido la plataforma elegida. Si por el motivo que fuera no alcanzáramos las especificaciones deseadas, las ideas generales de este trabajo se podrían aprovechar para realizar la implementación en otra basada en un procesador más potente, quizá un DSP.

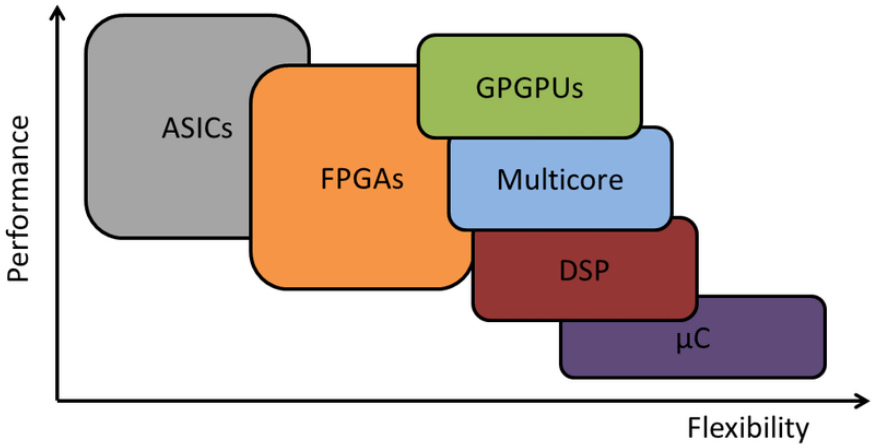



**Figura 1. Esquema de eficiencia y flexibilidad de distintos dispositivos digitales**

La idea de realizar un amplificador *lock-in* digital basado en un microcontrolador no es algo nuevo. Hace 20 años se presentó un trabajo en el que se empleaba un microcontrolador para implementar un amplificador lock-in de baja frecuencia en el que se cuidaba mucho la relación señal-ruido [11] y no se daba mucha importancia a la versatilidad que proporcionan los sistemas digitales reconfigurables. A modo de ejemplo, la frecuencia de corte de los filtros digitales no era configurable. Nuestro objetivo es aprovechar los avances de estos 20 años y centrarnos más en la parte digital para demostrar que un microcontrolador puede funcionar como un DSP en un amplificador lock-in de bajo coste y de calidad suficiente como para que su uso en entornos científicos y tecnológicos pueda ser una realidad. Para aplicaciones concretas donde se necesite uno de gama alta siempre existirán las opciones comerciales.

## CAPÍTULO 2. PROBLEMÁTICA Y ESTADO DEL ARTE

Los amplificadores lock-in son instrumentos que calculan de una manera rápida y con gran inmunidad a las interferencias o al ruido en general la amplitud de una señal sinusoidal y su desfase respecto a otra de referencia de la misma frecuencia. A diferencia de otras técnicas, como la transformada de Fourier, no adquieren bloques de datos y los analizan en su conjunto para obtener información de todas las frecuencias y luego seleccionar la de interés, sino que aplican cálculos sencillos para desplazar la señal en el dominio de las frecuencias y es sobre esta señal desplazada sobre la que se aplican los filtros, obteniendo un único valor, en lugar del espectro completo. El tiempo en obtener este valor depende principalmente del filtro empleado, que en general es digital de respuesta infinita a un impulso (*IIR, Infinite Impulse Response*) en la inmensa mayoría de los dispositivos comerciales, siendo lo habitual que se pueda configurar tanto la frecuencia de corte con el orden del filtro. La inmunidad al ruido, en cambio, sí depende de las etapas analógicas, por lo que es una especificación importante que se debe buscar en cuanto a la capacidad de procesado de señales. Otra es la frecuencia máxima que puede analizar y quizá la cuarta sea el número y combinaciones de señales a analizar de manera simultánea. Hemos indicado que, a diferencia de otras, esta técnica obtiene un único valor de amplitud y desfase, pero nada impide replicar ciertos bloques para realizar los mismos cálculos en varias unidades con distintos parámetros. Estos bloques se conocen como demoduladores y las versiones más avanzadas incorporan hasta 8, permitiendo analizar simultáneamente 2 señales y obtener información de hasta 4 frecuencias de cada una de ellas [12]. Una vez obtenidos los valores se deben almacenar o enviar a algún otro dispositivo. Esta comunicación puede ser analógica o digital, siendo en general la digital más adecuada para evitar que la señal se ensucie por el camino. Téngase en cuenta que algunos amplificadores anuncian valores de ruido tan pequeños como 4nV/√Hz, similar al ruido térmico de una resistencia de 1kΩ a temperatura ambiente. El problema de la comunicación digital es que no existe un estándar único, cada fabricante emplea su propio protocolo de comunicación y sus interfaces de programación (*API, Application Programming Interface*) en caso de disponer de ellas. Estas interfaces suelen permitir comunicar los dispositivos con programas escritos en C, Matlab o LabView y últimamente también Python.

La tendencia actual es implementar el procesado digital en dispositivos programables, principalmente FPGA. Las FPGA son muy apropiadas, ya que permiten controlar convertidores de digital a analógico (DAC) y de analógico a digital (ADC) de una manera relativamente sencilla, paralelización de código, operaciones matemáticas de manera muy eficiente, etc. Esto es posible gracias a la cantidad de puertas lógicas equivalentes, frecuencia de funcionamiento y a la gran cantidad de entradas y salidas configurables que poseen. El ancho de banda de las señales analizadas en implementaciones sobre FPGA puede ir desde cientos de kHz hasta cientos de MHz, con frecuencias de muestreo del orden de GSPS [12], [13]. El precio de tarifa va en consonancia con las especificaciones, siendo de varios miles de euros las versiones más básicas y decenas de miles de euros las más avanzadas. En la Figura 2 [12] mostramos un amplificador lock-in comercial de altas prestaciones junto con las características principales que indica el fabricante.


**Figura 2. Amplificador lock-in comercial de gama alta**

La Tabla 2-1 muestra una comparativa de las especificaciones principales de los dos amplificadores lock-in de gama más alta que hemos encontrado en el mercado actual. Los precios no se indican porque no son públicos, pero en ambos casos se han recibido ofertas de varias decenas de miles de euros.

**Tabla 2-1. Comparación de amplificadores lock-in de gama alta**

| Modelo | Ancho de banda (MHz) | Frecuencia de muestreo (GSa/s) | Densidad de ruido (nV/√Hz) | Interfaces |
|---|---|---|---|---|
| Zurich Instruments UHFLI | 600 | 1.8 | 4[1] | Analógica<br>Ethernet<br>USB |
| Liquid instruments Moku:Pro | 600 | 1.25 | 30 | Analógica<br>Registro de datos digital |

Este tipo de amplificadores no son comparables al presentado en este trabajo ni en precio, ni en especificaciones ni en tecnología del procesador digital de señales. Buscando en la literatura trabajos previos en los que se emplee un microcontrolador como núcleo digital de un amplificador lock-in solo hemos encontrado un artículo revisado por pares y es de hace 20 años [11], que por su antigüedad y la evolución de la tecnología en este tiempo no consideramos que forme parte del estado del arte actual. Existe también una entrada de un blog de 2018 [14] en la que el autor presenta varios tipos de amplificadores lock-in basados en microcontroladores con más o menos tareas asignadas a la parte digital. Entre ellos hay uno en el que el procesamiento de la señal sí se realiza en el microcontrolador, aunque para la generación de la señal de referencia emplea un sintetizador externo. El micro empleado es un PIC32MZ configurado para

[1] 4nV/√Hz para frecuencias superiores a 100kHz

funcionar a 240MHz de frecuencia de reloj y cuenta con una unidad de coma flotante de doble precisión implementada en *hardware*. La tasa de refresco de los convertidores de analógico a digital llega hasta 2MSPS (*Mega Samples Per Second*) en modo de 16 bits y hasta 2.4MSPS en modo de 8 bits, lo que tampoco implica necesariamente que se estén utilizando a estas frecuencias, si el resto de equipo no lo permite. El autor indica que el dispositivo es capaz de adquirir bloques de datos y realizar su FFT (*Fast Fourier Transform*) y en este caso sí detalla que muestrea a 2MSPS, pero en el caso de funcionar como amplificador lock-in, en el que se requieren más cálculos por ciclo, habla de reducir el ruido con sobremuestreos de 16:1, lo que nos hace pensar que la tasa real de funcionamiento en este modo sea de 125kSPS. El montaje final presenta dos entradas a analizar y una salida de excitación, no posee salidas de resultados en tiempo real y por tanto su uso queda limitado por el ancho de banda y tiempos de respuesta del bus USB, lo que reduce mucho sus posibilidades de uso reales. La Figura 3 muestra el modelo 3d de las placas y el montaje final puede verse en la Figura 4.

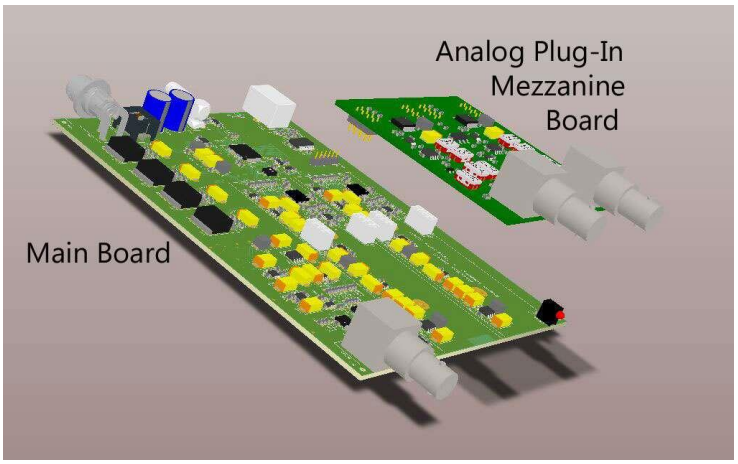



**Figura 3. Circuito de amplificador lock-in basado en microcontrolador**


**Figura 4. Amplificador lock-in basado en microcontrolador**

## CAPÍTULO 3. DISEÑO

En este capítulo nos vamos a centrar en el funcionamiento de los amplificadores lock-in digitales, los módulos que los forman y cómo se conectan entre ellos para realizar un diseño genérico que simularemos en el CAPÍTULO 4. Este diseño será la base para la implementación en un microcontrolador, que al ser un dispositivo programable permitirá realizar unas u otras operaciones sobre la misma implementación *hardware*.

### 3.1 Descripción funcional de los componentes

El amplificador lock-in emplea cálculos sencillos para desplazar la frecuencia de la señal a analizar y posteriormente filtrarla para mejorar su relación señal/ruido y eliminar componentes no deseadas en general. Al ser una herramienta que se lleva utilizando muchos años ha ido sufriendo modificaciones para adaptarse a ciertos usos típicos y las distintas implementaciones suelen incluir modos de funcionamiento más allá de los que proporcionaban las versiones iniciales. Si tenemos en cuenta estos modos que se han convertido en opciones habituales podemos diferenciar los bloques principales que se muestran en la Figura 5 y que constituyen el diseño final de un amplificador lock-in digital genérico. Los números entre paréntesis de los títulos de cada componente se corresponden con los del esquema.

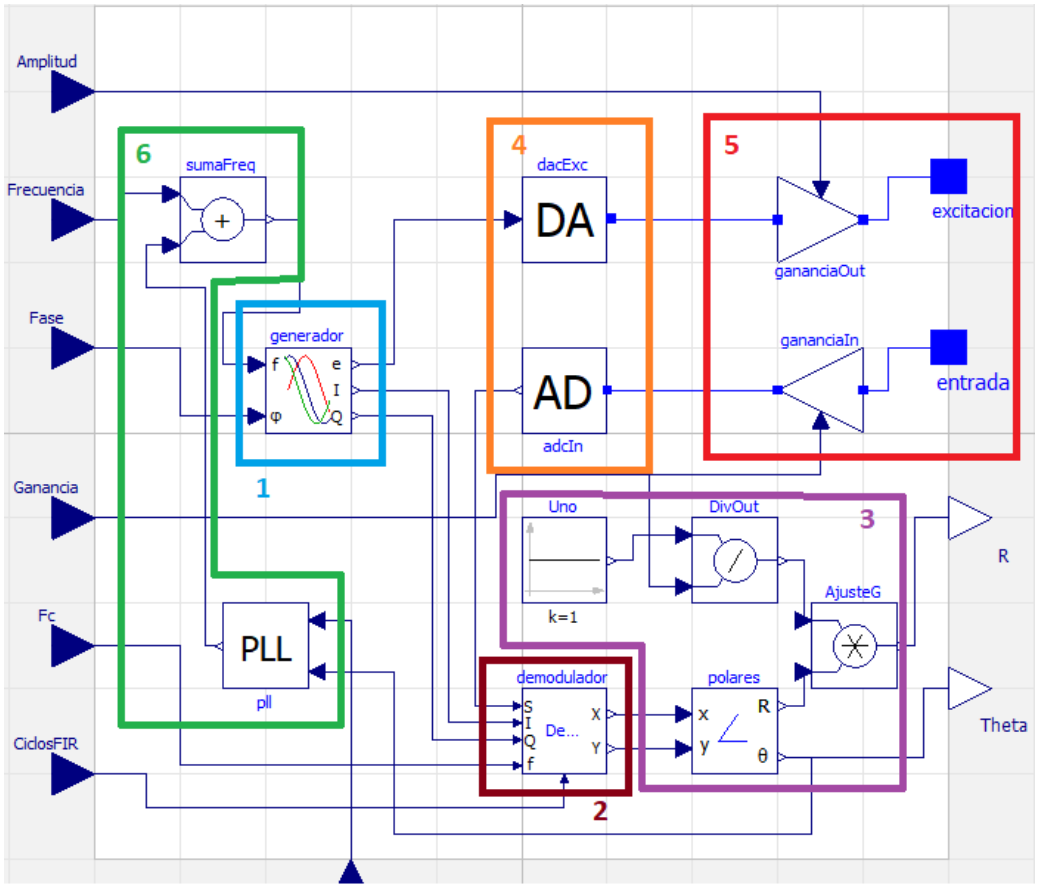



**Figura 5. Módulos del amplificador lock-in**

### 3.1.1 Generación de las señales de excitación y referencia (1)

Hemos indicado que el objetivo del amplificador lock-in es obtener la amplitud de una señal a analizar y su desfase respecto a otra de referencia de la misma frecuencia. Es evidente que para poder compararla con una referencia esa referencia debe existir, o al menos algunos parámetros que permitan reconstruirla. En la señal de entrada hay más libertad, porque en principio se puede analizar cualquier señal de corriente alterna (AC, *Alternating Current*). Lo habitual es que el propio amplificador incluya un generador de tensión AC configurable, conocido como señal de excitación, que se conecte al dispositivo a caracterizar o controlar. Si el sistema que se conecta al amplificador lock-in tiene una respuesta lineal a esta excitación, generará una señal de la misma frecuencia, variando su amplitud y desfase. Si no lo es, en general esta respuesta tendrá una descomposición en frecuencias múltiplos enteros de la de excitación (armónicos), a la que se suele llamar ω. Así, muchos amplificadores modernos permiten analizar la amplitud no solo de la frecuencia original, sino también de sus armónicos, señales que se deben generar en este mismo módulo.

Normalmente cuando se quiere conectar un dispositivo al amplificador lock-in hay que añadir algo de circuitería o elementos mecánicos, que introducen retardos en la propagación de las señales. La señal de excitación idealmente suele ser una tensión sinusoidal pura, normalmente de baja potencia, que puede amplificarse en corriente para manejar una bobina que genere un campo magnético, o ser la señal de control de un actuador mecánico que oscile una cierta pieza a una determinada frecuencia. Los retardos a veces son de tiempo constante y a veces dependientes de la frecuencia, en cualquier caso, interesará compensarlos. Es más, los propios cálculos y conversiones entre en dominio analógico y el digital incluirán otros retardos independientemente de los que puedan provenir del dispositivo controlado. La consecuencia es que en general habrá un retardo artificial entre la señal de excitación y la de entrada, e interesará poder compensarlo para medir los desfases de las señales de una manera precisa, y solo los desfases del sistema físico, sin tener en cuenta los instrumentales o incluso estudiar solo el desplazamiento de la fase desde un origen arbitrario. El módulo generador de señales debe ser capaz de controlar el desfase entre la señal de excitación y la de referencia.

### 3.1.2 Demodulador (2)

Es la parte principal que se encarga del análisis y descomposición de la señal. A partir de la entrada que se quiere analizar y los resultados que recibe del generador de las señales de excitación y referencia calcula un par de valores que serán las coordenadas de un vector cuyo módulo está relacionado con la amplitud de la señal analizada y el argumento con el desfase respecto a la de referencia. Debido a las fórmulas matemáticas que se emplean para realizar este análisis, el resultado se obtiene en coordenadas cartesianas, siendo más intuitivo para el usuario normalmente el uso de coordenadas polares. Si el amplificador lock-in incluye la conversión a polares hará falta otra etapa que realice esta conversión, que se puede considerar parte del demodulador o no. En nuestro diseño hemos considerado que esta conversión no forma parte del demodulador propiamente dicho.

### 3.1.3 Adaptación de los resultados para mostrarlos al usuario (3)

Si la conversión de coordenadas cartesianas a polares no se considera parte del demodulador, se debe considerar como parte de un módulo adicional que realice las operaciones necesarias para mostrar al usuario los resultados de los cálculos de la manera más sencilla y precisa posible. Este módulo debe incluir cualquier otro tipo de normalización necesaria, quizá permitir al usuario tener los datos como amplitud o voltaje eficaz o compensar las posibles ganancias de las etapas analógicas, que comentaremos más adelante.

### 3.1.4 Convertidores de Digital a Analógico y de Analógico a Digital (4)

En un sistema digital necesitamos convertir los números en tensiones (u otra magnitud física) y las tensiones leídas en números. Para el primer objetivo emplearemos un convertidor de digital a analógico, mientras que en el segundo necesitaremos un convertidor de analógico a digital.

### 3.1.5 Etapas analógicas de entrada y salida (5)

Se podría considerar que estas etapas pertenecen a la conversión propiamente dicha o que tienen entidad propia, dependiendo principalmente de su complejidad y la arquitectura elegida. Son etapas que pueden amplificar la señal analógica, filtrarla, adaptar impedancias, etc. En este diseño solo las vamos a emplear para ajustar los límites de tensión analógicos al fondo de escala del DAC o el ADC. Si, por ejemplo, la implementación final usa un DAC con una referencia de tensión externa, esta ganancia se debería incluir en la parte de conversión.

### 3.1.6 PLL (6)

El lazo de control de fase (*Phase Locked Loop*, PLL) es una de las opciones que comentábamos que no forma parte de un amplificador lock-in estrictamente hablando pero que en muchos entornos se ha popularizado tanto que es habitual que se incluya, al menos como opción. Se emplea en sistemas resonantes trabajando en su frecuencia natural. Sabemos de la teoría de control que si un sistema tiene algún par de polos complejos y su factor de amortiguamiento es lo suficientemente pequeño, el módulo de su función de transferencia tendrá un máximo a una determinada frecuencia, y que la fase a esa frecuencia será conocida, dependiendo del resto de polos y ceros. La fase que aporta este par de polos conjugados en la frecuencia de resonancia debe ser -90°. Desde el punto de vista puramente electrónico tiene ventajas trabajar en la frecuencia de resonancia, ya que empleamos el sistema físico como un amplificador o filtro pasa-banda de frecuencias, con lo que mejoramos la relación señal ruido. Pero esta frecuencia puede variar con el tiempo, debido a cambios en las propiedades de los elementos que forman el sistema físico o a su interacción con otros elementos. Si conseguimos mantener el sistema en su resonancia podemos, por una parte, compensar estos cambios de frecuencia, y por otra, medirlos y quizá obtener información cuantitativa sobre la perturbación que está ocasionando el cambio y caracterizarla, siendo en algunos

casos este el fin último. Su funcionamiento es sencillo, a frecuencias de trabajo por debajo de la resonancia la fase será más cercana a cero y por encima a -180°. Si aplicamos un lazo de realimentación sobre la fase, de manera que variemos la frecuencia añadiendo o restando una cierta cantidad, podremos mantener el sistema en resonancia. Para realizar este control podemos emplear -90° de valor de consigna si suponemos que la función de transferencia no tiene más polos o ceros, pero como esto es muy complicado de conseguir, en general usaremos algún valor que nos resulte más cómodo y variaremos el desfase de la señal de referencia para "colocar" la resonancia en la fase que nos interese.

## 3.2 Descripción matemática

Para describir matemáticamente el amplificador lock-in vamos a emplear el mismo diagrama de bloques de la Figura 5.

### 3.2.1 Generación de las señales de excitación y referencia

Aunque lo veremos con más detalle al modelar el demodulador, son necesarias una señal en fase con la que queramos analizar (*In-phase*, I) y otra desfasada 90º, que se suele denominar en cuadratura (*Quadrature*, Q), aparte de la señal de excitación (exc) que ya se discutió que debe poder tener un desfase arbitrario (φ). Además de lo anterior interesa que la señal sea continua ante cambios de la frecuencia de funcionamiento, ya sean estos cambios por petición del usuario o por efecto del PLL, buscando mantener el sistema en su resonancia. Para conseguir esta continuidad una opción es generar una función lineal que será la fase de la señal "I" (θ) y a partir de ella calcular las fases de las demás y el valor de su seno. Esta función fase variará según la frecuencia (f) a generar, teniendo que aumentar una vuelta completa (2π radianes) en cada ciclo. Obtenemos por tanto las primeras ecuaciones del modelo, siendo t el instante de tiempo actual:

$$\frac{d\theta}{dt} = 2 \cdot \pi \cdot f$$
$$I = sen(\theta)$$
$$Q = sen(\theta + \pi/2)$$
$$exc = sen(\theta + \varphi)$$

El sistema de simulación calculará numéricamente el valor de θ en cada ciclo de la simulación (n) a partir de su derivada. En el caso más sencillo, empleando el método de Euler, será:

$$\theta(n) = \theta(n-1) + \frac{d\theta}{dt} \cdot \Delta t$$

siendo Δt la diferencia de tiempo entre el instante de simulación actual y el anterior. Algoritmos más complejos requerirán expresiones con más términos.

### 3.2.2 Demodulador

El primer paso para obtener la amplitud y el desfase de dos señales sinusoidales de la misma frecuencia consiste en "desplazar" la frecuencia aplicando igualdades trigonométricas, en concreto, para unos ángulos genéricos α y β:

$$sen(\alpha) \cdot sen(\beta) = \frac{\cos(\alpha - \beta) - \cos(\alpha + \beta)}{2}$$
$$sen(\alpha) \cdot cos(\beta) = \frac{\mathrm{sen}(\alpha + \beta) + \mathrm{sen}(\alpha - \beta)}{2}$$

Ambas fórmulas descomponen las señales en otras de dos frecuencias distintas, siendo estas la suma y la diferencia de las originales. Por simplicidad vamos a suponer que la frecuencia de trabajo del amplificador lock-in fuera constante, que significaría que el usuario no ha modificado los parámetros durante un determinado tiempo y que el PLL está desactivado, y vamos a considerar el origen temporal de manera que inicialmente la fase del generador de señales sea nula. Podemos reformular I y Q en función de la pulsación de la señal generada (ω) como:

$$I = sen(\omega \cdot t)$$
$$Q = sen(\omega \cdot t + \pi/2) = \cos(\omega \cdot t)$$

Si también suponemos que la respuesta del sistema es de la misma frecuencia, con amplitud A y un desfase arbitrario $\varphi$ respecto a la señal "I", su expresión sería:

$$f(t) = A \cdot sen(\omega \cdot t + \varphi)$$

Por tanto, tenemos que:

$$f(t) \cdot I = A \cdot sen(\omega \cdot t + \varphi) \cdot sen(\omega \cdot t) = \frac{\mathrm{A} \cdot \cos(\varphi)}{2} - \frac{A \cdot \cos(2 \cdot \omega \cdot t + \varphi)}{2}$$
$$f(t) \cdot Q = A \cdot sen(\omega \cdot t + \varphi) \cdot cos(\omega \cdot t) = \frac{\mathrm{A} \cdot \mathrm{sen}(\varphi)}{2} + \frac{A \cdot \mathrm{sen}(2 \cdot \omega \cdot t + \varphi)}{2}$$

Con cada una de estas operaciones obtenemos una señal constante y otra del doble de frecuencia que la original. El resto de las componentes de f(t) también sufrirán un desplazamiento equivalente. Por ejemplo, si f(t) tiene componente de corriente continua no nula, algo muy habitual, la veremos como un seno de la frecuencia de funcionamiento.

Si somos capaces de eliminar las componentes de alta frecuencia obtendremos dos valores que no dependen del tiempo y están relacionados con la amplitud y el desfase como si fueran las coordenadas cartesianas de un vector. Tradicionalmente se han empleado filtros analógicos pasa-baja para eliminar estas altas frecuencias, que en implementaciones más modernas se han sustituido por filtros digitales, normalmente

equivalentes, de tipo IIR. Suponiendo ideales estos filtros que denotaremos como "filtro()", de manera que solo dejen pasar la señal de continua, podemos definir unas nuevas variables x e y como:

$$x = \frac{A}{2}cos(\varphi) = filtro(f(t) \cdot I)$$
$$y = \frac{A}{2}sen(\varphi) = filtro(f(t) \cdot Q)$$

Estos valores equivalen a las coordenadas cartesianas de un vector cuyo módulo es la mitad de la amplitud de la señal medida y el argumento el desfase buscado.

#### 3.2.2.1 Filtros digitales

En nuestro diseño hemos incluido el filtrado de las señales dentro del propio demodulador, aunque se podría considerar un módulo independiente. Ya indicamos que en general las versiones digitales heredan muchas funcionalidades de las analógicas, y el filtrado de los cálculos intermedios es una de ellas, por lo que la mayoría de las versiones comerciales emplean filtros tipo IIR. Basta mirar las hojas de características para darse cuenta de que indican la atenuación por encima de la frecuencia de corte en decibelios por octava, algo característico de estas implementaciones, tanto de los filtros IIR como de los analógicos [12], [15].

Recordemos que el objetivo principal del filtrado es quedarnos solo con la componente de continua, reduciendo así tanto el ruido de la señal a analizar como la magnitud de los artefactos creados por la técnica de desplazamiento de frecuencias, que son siempre armónicos de la frecuencia de trabajo. Realmente la frase anterior es matizable, sería así si quisiéramos hacer medidas estáticas, que normalmente no es el caso. Decíamos que una de las ventajas de los amplificadores lock-in es que proporcionan medidas rápidas, por lo que si el tiempo de respuesta no fuera importante quizá serían más convenientes otras técnicas, como la transformada de Fourier o la transformada *wavelet*. El tiempo de respuesta de un filtro IIR es de varias constantes de tiempo ($\tau$). Según el orden y el intervalo de estabilización que consideremos podemos decir que son entre 3 y 6 constantes de tiempo, típicamente. Para hacernos una idea de órdenes de magnitud, la frecuencia de corte es $2 \cdot \pi \cdot \tau$, es decir, del mismo orden que el tiempo de estabilización.

Hemos visto que como resultado del producto de la señal a analizar por las señales I y Q siempre tendremos una componente en el doble de la frecuencia de trabajo de una magnitud similar a la de continua y si la señal de entrada no está centrada en 0 también tendremos otra componente en la propia frecuencia $\omega$. Para filtrar estas frecuencias de manera eficiente y obtener tiempos de respuesta rápidos, necesitamos una vez más buscar un compromiso al configurar la frecuencia de corte o la constante de tiempo de filtrado. Valores mayores de la constante de tiempo producirán señales más limpias, pero tardarán más en obtener los resultados, y valores más pequeños, con respuestas más rápidas, pueden no llegar a eliminar satisfactoriamente los artefactos.

En implementaciones analógicas se suelen emplear baterías de filtros de primer o como mucho segundo orden, de frecuencia fija y el usuario elige cuál activa en cada momento. En las digitales este problema es más sencillo de resolver, ya que se pueden recalcular los coeficientes en tiempo de ejecución.

Si empleamos por ejemplo un filtro de segundo orden (40 dB/década o 12db/octava) y queremos reducir la magnitud de una componente al 1% de su valor inicial necesitamos una frecuencia de corte 10 veces inferior, es decir, $0.1\omega$. Esto se traduce en un tiempo de estabilización aproximado de 10 ciclos de oscilación. Variando el orden del filtro estos valores también cambian, aunque un filtro de segundo orden en general es una buena opción. Uno de primer orden nos obliga a bajar mucho la frecuencia de corte para conseguir un buen filtrado y los de órdenes muy altos pueden presentar problemas de precisión en implementaciones reales. Podemos adelantar, aunque lo comentaremos con más detalle después, que incluso uno de segundo orden ha necesitado un cuidado especial en su implementación en el microcontrolador para evitar estos problemas al trabajar con constantes de tiempo relativamente altas.

Una manera sencilla de eliminar ciertas componentes y sus armónicos y obtener tiempos de respuesta pequeños consiste en emplear filtros de respuesta finita ante un impulso. De entre todos ellos, el más sencillo consiste en emplear coeficientes del mismo peso para todos los factores, o lo que es lo mismo, calcular el promedio de una señal en un intervalo de tiempo y quizá aplicarle una ganancia. Sin entrar en tecnicismos ni en demostraciones matemáticas, es fácil observar que el valor promedio de una señal sinusoidal en un periodo es nulo, por lo que si empleamos de intervalo para el promedio exactamente un ciclo de la frecuencia a analizar anularemos el valor de esa componente. Este razonamiento, cierto para señales continuas, puede dejar de serlo si son muestreadas. Si el semiperiodo de la señal es múltiplo entero del de muestreo, cada valor positivo tendrá uno equivalente negativo, se anularán y el resultado del promedio será cero, pero si no lo es, el resultado será un valor no nulo. Tal y como nos dice la intuición, en general cuantos más puntos tengamos para promediar en un ciclo, menor será el resultado del promedio y mejor filtrará las componentes no deseadas, siempre con mejor o peor resultado en función de la proporción exacta entre la frecuencia de muestreo y la que queremos eliminar. Otra ventaja de este filtro es que si el promedio de un ciclo es nulo también lo es el de dos, por lo que, si lo configuramos para eliminar, o al menos reducir, la componente en $\omega$, conseguiremos el mismo efecto o parecido en $2\omega$. Esta idea tan sencilla no suele implementarse en amplificadores comerciales, pero sí aparece en la literatura [16] y puede implementarse junto con los filtros de tipo IIR.

En nuestro caso simularemos los dos filtros en cascada, primero el IIR y luego el FIR aplicado sobre la salida del anterior. El IIR será de tipo Butterworth, que tiene la ventaja de presentar respuesta plana a bajas frecuencias hasta que empieza a caer su ganancia, lo que lo convierte en un filtro muy usado en general. Como indicamos antes, un filtro de segundo orden en general es un buen compromiso entre la capacidad de amortiguación de frecuencias altas y precisión numérica en los cálculos, por lo que, salvo que las simulaciones digan lo contrario, será el que empleemos.

### 3.2.3 Adaptación de los resultados para mostrarlos al usuario

El resultado del demodulador ya define la amplitud y fase de la señal analizada, que es el objetivo principal, pero para facilitar al usuario la interpretación de los datos es más adecuado indicar explícitamente estos valores a partir de x e y:

$$A = 2 \cdot \sqrt{x^2 + y^2}$$
$$\varphi = \operatorname{atan}\left(\frac{y}{x}\right)$$

Históricamente los amplificadores lock-in suelen llamar a estas magnitudes R y $\theta$ respectivamente, como puede verse en la Figura 6 [17]. El valor de R se puede expresar como amplitud, valor eficaz, porcentaje sobre la señal de referencia, o cualquier otra normalización.


**Figura 6. Interfaz de usuario en un amplificador lock-in comercial**

La señal analógica de entrada se amplifica por un valor constante antes de alimentar el ADC para ajustarse mejor a su fondo de escala y reducir el error de cuantificación. Esta ganancia (G), que suele ser variable y según la implementación puede elegirla el usuario o el sistema, puede compensarse en los datos que se muestren, de manera que el valor que se indique al usuario (R') sea el de antes de amplificar.

$$R' = \frac{R}{G}$$

### 3.2.4 Convertidores de Digital a Analógico y de Analógico a Digital

El paso del dominio analógico al digital y viceversa se realiza mediante convertidores de tensión de un ancho de palabra que será una característica del propio convertidor, junto con su fondo de escala. Para las simulaciones que realizaremos vamos

a suponer un rango de funcionamiento de ±1V, por lo que si el ancho de palabra es m bits el paso de cuantización (LSB) será:

$$LSB = \frac{1-(-1)}{2^m} = \frac{1}{2^{m-1}}$$

Si el valor a convertir supera el intervalo de funcionamiento el resultado será el límite de este intervalo y en caso contrario será el múltiplo del valor LSB más cercano al pedido para la conversión.

### 3.2.5 Etapas analógicas de entrada y salida

Las etapas analógicas pueden incluir distintos tipos de adaptación de la señal. Para el diseño simplemente hemos supuesto unos amplificadores de tensión para ajustar las señales al fondo de escala de los convertidores. La tensión de salida ($V_{out}$) es proporcional a la de entrada ($V_{in}$) pudiéndose configurar la constante de proporcionalidad (G).

$$V_{out} = G \cdot V_{in}$$

### 3.2.6 PLL

El lazo de realimentación que mantiene el sistema en resonancia puede tener distintas arquitecturas, siendo los controladores PI los más utilizados. Para las simulaciones supondremos un controlador discreto y la acción de control se sumará a la frecuencia de funcionamiento, es decir, en realidad calcula el desplazamiento de frecuencia respecto a la que seleccionó el usuario al empezar el experimento ($\Delta frec$). Su activación debe ser opcional.

Existen distintas implementaciones con algunos matices en cuanto a los parámetros de configuración y las fórmulas exactas. En nuestro caso los parámetros serán el valor objetivo de la variable controlada ($consigna$), la constante proporcional ($Kp$) y el tiempo de integración ($T_i$). Definimos una variable auxiliar ($error$) que almacena la diferencia entre el valor de fase deseado y el leído ($\theta$). El término integral ($I$) equivale a la integral numérica del error acumulado, empleando rectángulos para obtener el valor en un ciclo n a partir del error y el intervalo de tiempo transcurrido desde el ciclo anterior ($\Delta t$). Las consideraciones anteriores conducen a la siguiente definición matemática cuando el controlador está activo:

$$error = consigna - \theta$$
$$I(n) = I(n-1) + error \cdot \Delta t$$
$$\Delta frec = Kp \cdot \left( error + \frac{1}{T_i} I \right)$$

# CAPÍTULO 4. SIMULACIONES

Como parte de este trabajo se ha desarrollado una biblioteca en el lenguaje Modelica para simular distintas configuraciones de amplificadores lock-in. En este capítulo comentaremos algunos detalles del lenguaje y entorno empleados, hipótesis de modelado, detalles de implementación y resultados de las simulaciones.

## 4.1 Tecnología empleada

Modelica [18] es un lenguaje para el modelado físico cuyo desarrollo comenzó en 1996, siendo la primera versión de 1997. El objetivo principal del equipo internacional que lo creó era unificar las tendencias de la época en lo relativo a modelado y simulación de sistemas físicos [19]. La entidad que se encarga de su desarrollo, mantenimiento y promoción es la Modelica Association, una organización sin ánimo de lucro con miembros de Europa, Norteamérica y Asia. A diferencia de otros paradigmas de modelado previos, en el físico los modelos están formados por distintos componentes que simulan el comportamiento natural de ciertos elementos básicos y los sistemas más complejos se forman mediante la composición de estos. Es una descripción más potente, sencilla y natural que por ejemplo el modelado basado en diagramas de bloques, en el que la causalidad es fija, algo artificial en muchos casos. En el modelado físico no se definen entradas y salidas sino conexiones sin una dirección predeterminada y es el sistema de simulación el que analiza las ecuaciones y decide en qué orden deben ejecutarse, qué variable se debe calcular de qué ecuación, etc. Yendo un paso más lejos, en sistemas híbridos en los que un cambio de estado provoca que cambie la estructura del modelo, y con ella la causalidad de las ecuaciones, esta asignación puede variar a lo largo de la simulación. Modelica permite dos tipos de conexiones, según se deba cumplir que el valor de todos los elementos conectados sea el mismo (por ejemplo, la tensión eléctrica en un nodo de un circuito) o que su suma sea nula (la corriente eléctrica en el nodo). Cada elemento del modelo se puede definir mediante ecuaciones diferenciales algebraicas (EDA), ecuaciones diferenciales ordinarias (EDO), *bond graphs*, autómatas finitos, etc. Para el caso de sistemas híbridos, se pueden definir eventos que modifiquen la dinámica del sistema o los valores de las variables de estado.

Otro de los fundamentos de Modelica son los modelos parciales y la herencia, que facilitan la reutilización de código. Los modelos parciales son definiciones de estructuras comunes que compartirán ciertos elementos pero que por sí solas no definen el comportamiento final y tienen que ser completadas mediante herencia, similares a lo que en otros lenguajes son interfaces o clases abstractas. Un ejemplo serían componentes electrónicos con dos conexiones, como condensadores, resistencias, bobinas, diodos, etc. Definiendo un modelo parcial que contenga dos conexiones eléctricas tendremos la base de todos estos componentes y cada uno de ellos añadirá la relación entre intensidad eléctrica y caída de tensión en sus bornes. Esta facilidad para la reutilización, unida al espíritu de proporcionar una base sólida sobre la que construir un lenguaje estándar, se concretan en distintas librerías disponibles con gran cantidad de modelos ya implementados, que se pueden consultar, y en algunos casos descargar, en la página de la Modelica Association [20]. Hay librerías de pago y gratuitas, desarrolladas por

instituciones públicas o privadas y por particulares, con distintos tipos de licencias, etc. Destaca la *Modelica Standard Library* (MSL), que es de código abierto y desarrollada por la Modelica Association. Mediante librerías como esta que presentamos contribuimos a avanzar en este estándar y si las dejamos disponibles públicamente a facilitar el trabajo a otros usuarios, que podrán utilizarlas y ampliarlas según sus necesidades.

Los programas de simulación que emplean Modelica permiten el tratamiento simbólico de ecuaciones y expresiones matemáticas en general. Salvo que el usuario indique lo contrario, el sistema de simulación será el encargado no solo de resolver los sistemas de EDO o EDA sino también de asignar la causalidad computacional, elegir las variables de estado e incluso eliminar los lazos algebraicos y reducir el orden del sistema, lo que facilita enormemente el proceso de modelado y simulación. Para este trabajo hemos elegido OpenModelica [21]. Es un programa gratuito que incluye un entorno de desarrollo integrado en el que los modelos se pueden definir gráficamente o mediante instrucciones de texto, incorpora las librerías MSL, permite realizar simulaciones y representar gráficamente los resultados, genera los ficheros de resultados en formato Matlab, etc.

## 4.2 Hipótesis de modelado

La biblioteca permite la construcción de distintos modelos, más o menos complejos, de amplificadores lock-in digitales. Un aspecto común a los distintos campos de la ingeniería, entre los que se encuentran los sistemas digitales, es la búsqueda del equilibrio entre costes y resultados para cumplir unas especificaciones, de manera que haya proyectos en los que el coste de fabricación sea un condicionante importante y se deban emplear componentes de bajo precio y otros en los que, por ser lotes pequeños o necesitar cumplir unas especificaciones más exigentes, no importe tanto el coste del material y se busquen otras características como la velocidad de cálculo, precisión, facilidad de desarrollo, etc. En cuanto a precisión, distintos procesadores de señales emplean distintas representaciones, como pueden ser la coma fija o flotante y distintos anchos de palabra, que están relacionados con el rango de valores admitidos y la precisión de la representación de los datos y del resultado de las operaciones matemáticas. Se ha supuesto que la representación y ancho de palabra del procesador simulado serán los mismos que los del entorno de simulación, aunque en general suele ser menor en sistemas en tiempo real como el que nos ocupa.

En lo relativo al dominio temporal, suponemos un reloj global para los distintos elementos que emplean señales muestreadas, como son los convertidores de digital a analógico y de analógico a digital, filtros digitales, etc. Este reloj se ha supuesto ideal, cada intervalo temporal durará exactamente el mismo tiempo independientemente de temperaturas, tensiones de alimentación, etc. Los retardos de propagación de las señales digitales se consideran nulos y las conversiones entre los dominios analógico y digital serán instantáneas. El DAC se comportará como un retenedor de orden 0 ideal, manteniendo constante el valor convertido hasta que se reciba el siguiente dato.

Los componentes analógicos se consideran ideales en general: fuentes de potencia infinita, parámetros no dependientes de temperatura u otros factores, condensadores e inductancias puros, sin ruido analógico de ningún tipo, sin retardos en la propagación, etc. La curva característica de los diodos se supone que sigue la ecuación de Shockley. Ya se indicó que una de las ventajas de los amplificadores lock-in es su buen rechazo al ruido, por lo que también interesa simularlo. Se ha modelado como una fuente de tensión de ruido de densidad espectral de potencia uniforme, lo que se suele conocer como ruido blanco. La distribución de ruido a alta frecuencia de la mayoría de los componentes electrónicos se suele aproximar a esta.

Los DAC y ADC se suponen perfectamente lineales y de ancho de banda infinito. En el caso del DAC la impedancia de salida se considera nula y la fuente de tensión de potencia infinita, mientras que en el ADC se considera una impedancia de entrada infinita. El amplificador de tensión también se supone con impedancia de entrada infinita y nula de salida. En ambos tipos de convertidores se ha supuesto un rango de funcionamiento analógico de ±1V.

## 4.3 Descripción de la biblioteca

Siguiendo el estándar empleado en las bibliotecas escritas en Modelica, hemos dividido la nuestra en 3 módulos (*packages*): Interfaces, Componentes y Ejemplos. También hemos añadido anotaciones gráficas, que son instrucciones que permiten asociar representaciones gráficas tipo icono a cada elemento, definir puntos de conexión y otras facilidades para poder componer los sistemas de manera gráfica sin conocer los detalles del lenguaje. Los módulos en que se divide el código empleado para las simulaciones son:

### 4.3.1 Interfaces

Contiene las definiciones de los conectores y bloques en los que se basarán los componentes, en algunos casos heredando de ellos y en otros creando objetos de estos tipos. En la parte digital las conexiones entre componentes en general tienen una direccionalidad bien definida, por lo que se han empleado conectores de entrada y salida y bloques con distintas opciones de estos conectores. Por simplicidad todas las conexiones se realizan con datos de tipo real, aunque el lenguaje permite otros tipos.

En la parte analógica, que se emplean magnitudes y componentes eléctricos, se ha definido un conector “Pin” que modela una conexión eléctrica en la que la tensión es la misma en todos los elementos de la unión y la suma de las intensidades de entrada (o salida) es nula. También se ha creado un modelo parcial que define dos pines, uno positivo y otro negativo y la tensión que cae en el elemento. Los distintos modelos de componentes eléctricos heredarán de esta clase.

### 4.3.2 Componentes

Para facilitar su uso, todos los componentes disponen de anotaciones gráficas que además permiten desarrollar los modelos compuestos sin tener que tocar código fuente, simplemente usando el editor gráfico que proporciona el entorno de simulación. Para conseguirlo se han incluido como bloques todas las funciones que se usan, por ejemplo, el cálculo del seno de un número, así como componentes que generan valores constantes o que varían linealmente con el tiempo. Esto hace que los diagramas sean algo más extensos frente a la opción de incluir estos valores como parámetros de cada elemento, pero a su vez que tengamos más información visual, lo que facilita la comprensión. Algunos de estos bloques pueden considerarse casos particulares de otros. Por ejemplo, uno genera rectas que dependen del tiempo y otro, valores constantes, que equivalen a rectas de pendiente nula. Se ha preferido este diseño, aunque incluya más componentes, para facilitar la interpretación de los diagramas. Los componentes incluidos en la librería se recogen en la Tabla 4-1.

**Tabla 4-1. Componentes de la biblioteca desarrollada**

| Nombre | Descripción |
|---|---|
| Constante | Genera una señal que mantiene su valor durante el tiempo. Se puede elegir el valor a generar. |
| Rampa | Genera un valor que varía linealmente con el tiempo. Se puede elegir el valor inicial (n) y la derivada respecto al tiempo (m). |
| Ganancia | Multiplica el valor de entrada por una constante. |
| Sin | Calcula el seno del valor de entrada. |
| Suma | Suma dos valores. |
| Producto | Multiplica dos valores. |
| Division | Divide un valor entre otro. |
| Temporizador | Define el periodo de muestro del sistema. Existirá un componente de esta clase, de manera que los elementos digitales que tengan que realizar una determinada tarea cada cierto tiempo sepan qué periodo de tiempo emplear. Sería algo similar a un reloj digital, una señal de muestreo para un ADC, de actualización para un DAC, etc. |
| Butter2LP | Aplica un filtro digital pasa baja a la señal de entrada. El filtro aplicado es de tipo Butterworth de segundo orden. La frecuencia de corte del filtro se define como una entrada, por lo que puede variar durante el experimento. |
| PromedioCiclos | Aplica un filtro de respuesta finita ante un impulso consistente en promediar los datos de un número determinado de ciclos de la señal de referencia. Considera que el cambio de un ciclo a otro se produce cuando la referencia pasa por cero en sentido ascendente. Si el número de ciclos que se indica es menor que 1 simplemente copia la entrada a la salida. |
| GenFase | Genera la señal de fase que se usará como entrada en la función seno para generar una señal suave de frecuencia variable, que se indica mediante una entrada. Inicialmente la salida de la fase vale 0. |

| Nombre | Descripción |
|---|---|
| ADC | Convertidor de analógico a digital. Lee un voltaje y genera un número que indica el valor leído, con la resolución correspondiente al ancho de palabra que se le indique. La tasa de refresco es la del temporizador global. |
| DAC | Convertidor de digital a analógico. Genera un voltaje igual al número que recibe, con la resolución correspondiente al ancho de palabra que se le indique. La tasa de refresco es la del temporizador global. |
| FuenteVcte | Fuente de tensión constante. Genera una tensión continua igual al valor de su entrada. |
| FuenteVruido | Fuente de tensión que genera valores pseudoaleatorios con una distribución de ruido blanco. Se puede configurar la potencia del ruido generado. |
| Resistencia | Resistencia eléctrica, la caída de tensión es proporcional a la corriente que la atraviesa. |
| Condensador | Condensador eléctrico, la corriente es proporcional a la derivada de la tensión respecto del tiempo. |
| Bobina | Inductancia eléctrica, la caída de tensión es proporcional a la derivada de la corriente eléctrica respecto del tiempo. |
| Diodo | Diodo cuya curva característica equivale a la ecuación de Shockley. |
| Masa | Referencia eléctrica donde la tensión vale 0V. |
| GananciaV | Amplificador de tensión controlado por una entrada numérica. La tensión de salida equivale al producto de la de entrada por el valor que se le indique. |
| Generador | Genera, como números, las señales sinusoidales que se emplearán en el amplificador lock-in, tanto las referencias como la señal de excitación que se enviará al DAC para alimentar el dispositivo a analizar o controlar. El desfase de esta última señal respecto a la referencia se indica en grados. Incluye un generador de fase que se encarga de que las señales sean continuas. |
| Demodulador | Recibe la señal a analizar y las referencias y calcula las componentes x e y del vector que indica la amplitud y fase de la señal analizada. Incluye filtros pasa-baja de segundo orden y filtros promedio, cuyos parámetros se pueden elegir. |
| Polares | Convierte coordenadas cartesianas a polares. En la biblioteca se usa para mostrar al usuario resultados en un formato más fácil de interpretar. |
| PLL | Controlador PI que mantiene el sistema en resonancia haciendo que la señal de fase se anule y modificando la frecuencia de trabajo. Incluye una señal de activación que cuando es menor o igual a cero desactiva el componente, anulando su salida. |
| LockinBasico | Modelo configurado empleando algunos de los componentes anteriores en el que se usa una configuración básica con parámetros preestablecidos o que son función de otros, para que usuarios no experimentados puedan realizar simulaciones sencillas. Su diagrama de bloques se muestra en la Figura 7. Se han empleado |

| Nombre | Descripción |
|---|---|
| | líneas de colores en las conexiones para facilitar la visualización. El usuario de la librería solo debe indicar la amplitud de la señal de excitación, su frecuencia y la ganancia de entrada y conectar el circuito eléctrico a analizar y el modelo ajusta los demás parámetros. No incluye PLL. |
| LockinCompleto | Al igual que LockinBasico es un modelo configurado listo para usar, pero en este caso tiene más opciones de configuración, lo que le aporta una mayor potencia para simulaciones avanzadas y a la vez requiere de mayor conocimiento para configurar adecuadamente todos los parámetros. Aparte de las opciones de la versión básica incluye un PLL cuya salida se suma a la frecuencia elegida y tiene más opciones de configuración, como son la frecuencia de corte de los filtros pasa-baja de tipo IIR, el número de ciclos a promediar para el filtro FIR y el desfase de la señal de excitación. Se puede ver su diagrama de bloques en la Figura 8. |

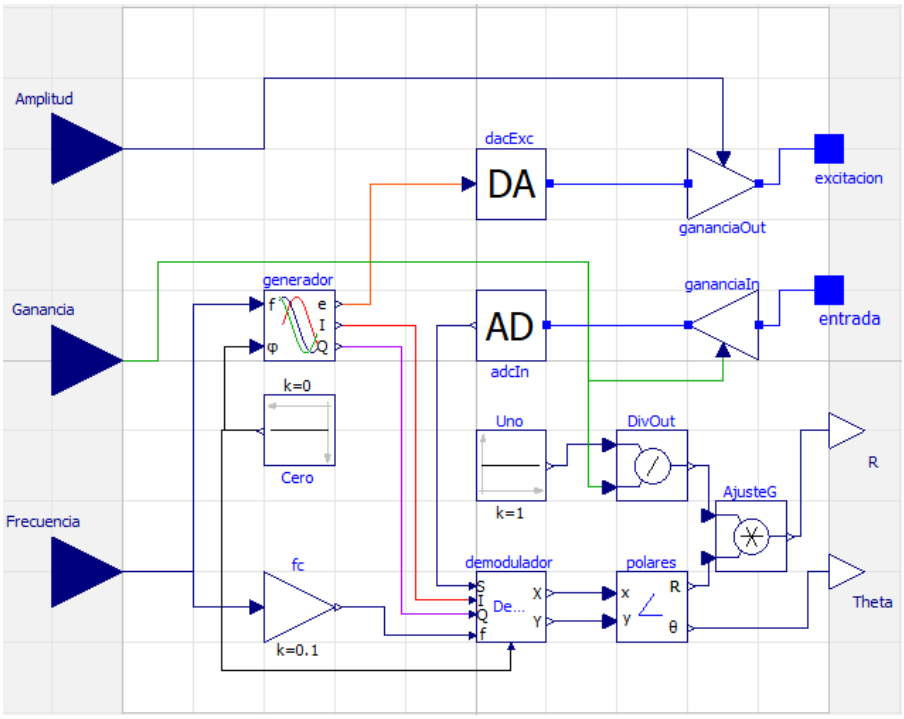



**Figura 7. Componente LockinBasico**

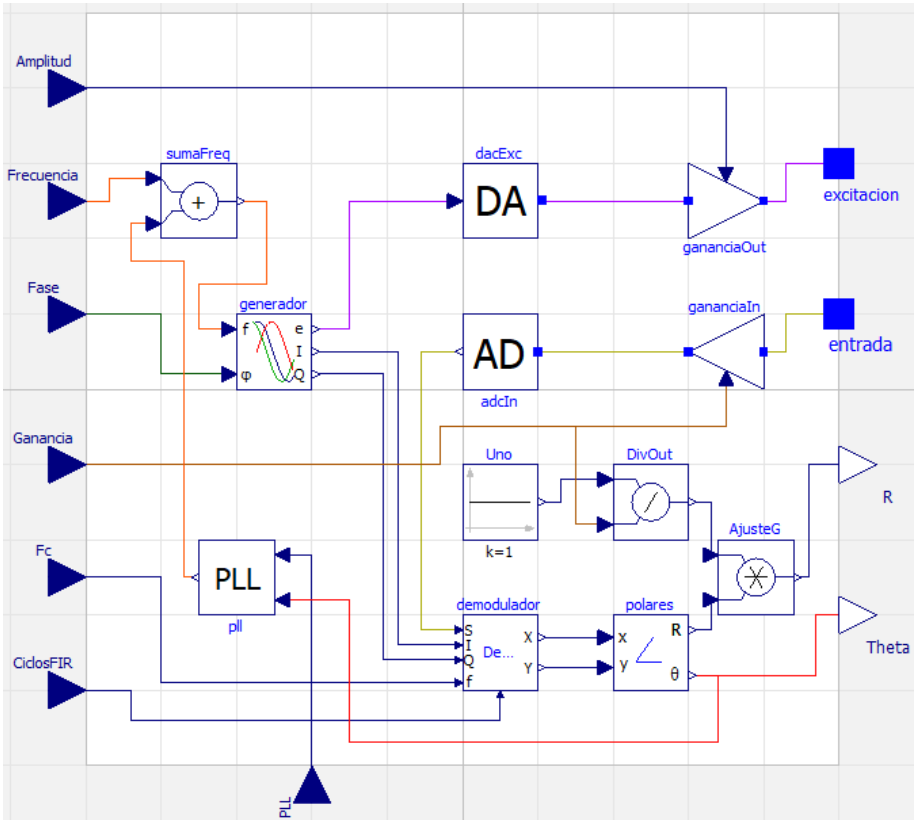



**Figura 8. Componente LockinCompleto**

### 4.3.3 Ejemplos

La librería incluye varios ejemplos para verificar el correcto funcionamiento de los componentes. El apartado 4.4 proporciona más información sobre cada uno de ellos y los resultados obtenidos.

## 4.4 Validación

La librería Lockin se ha validado usando el programa OMEdit de OpenModelica en su versión 1.18.1 y la librería MSL 4.0.0. Para la validación se han desarrollado los siguientes ejemplos que sirven de prueba de los componentes. Cada uno incluye una figura con su diagrama de bloques.

### 4.4.1 FiltroGenerador

Ejemplo sencillo que verifica el correcto funcionamiento del generador de ondas sinusoidales y el filtro pasa baja de segundo orden (Figura 9). Establece el temporizador de muestreo en 1ms, que será el periodo con el que se genere cada muestra digital, y varía la frecuencia del generador entre 0 y 100Hz. Se establece la frecuencia de corte del filtro en 45Hz, frecuencia a la que la ganancia deberá ser aproximadamente 0.7. Se puede ver en la Figura 10 que la señal filtrada mantiene su amplitud a frecuencias bajas y empieza a caer cerca de la frecuencia de corte, reduciendo aún más sus valores

cuando la supera. La escala temporal en segundos coindice con la frecuencia de la señal generada. La Figura 11 muestra un detalle en el entorno de la frecuencia de corte en el que se aprecia que la amplitud es la que debe, así como el efecto de la discretización del filtro, que actualiza su salida con el periodo de 1ms solicitado.

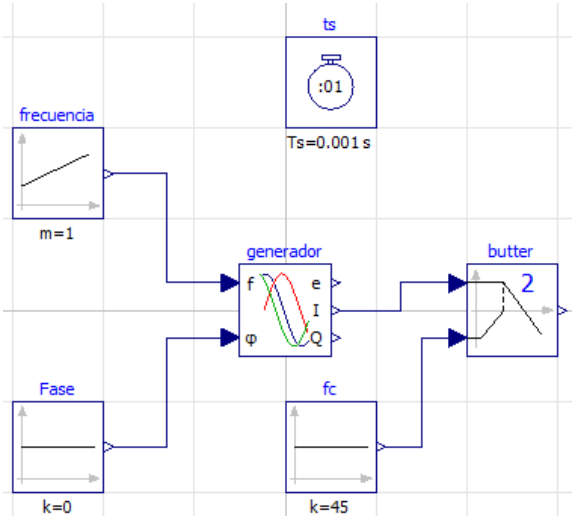



**Figura 9. Ejemplo FiltroGenerador**

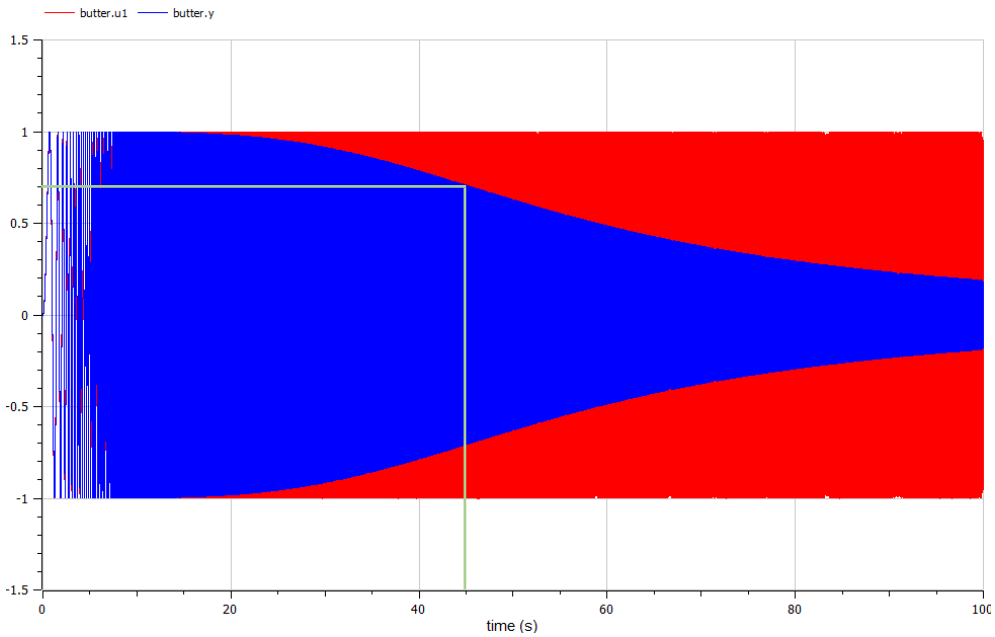



**Figura 10. Reducción de la amplitud al filtrar la señal**

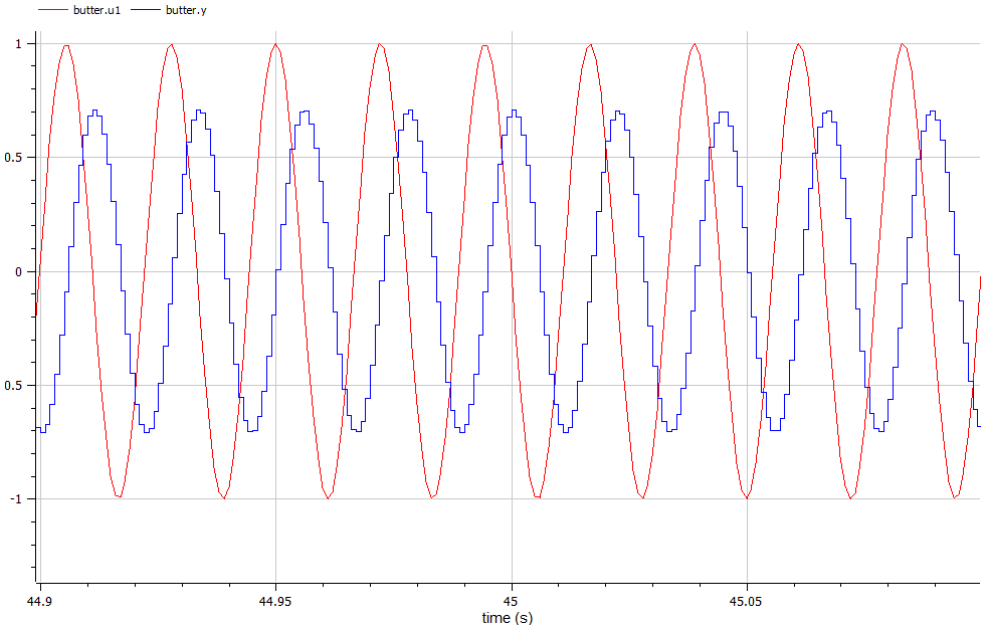



**Figura 11. Detalle en el entorno de la frecuencia de corte**

### 4.4.2 DemodulaTest

Prueba del demodulador, en este caso con periodo de muestreo de 1µs y con el sistema configurado para trabajar con señales de 10kHz, filtrando la salida a 1kHz. En la simulación se puede ver que el filtro elimina el rizado casi por completo, pero a cambio añade un retardo en la actualización de los resultados, que tardan algo más de 0.5ms en estabilizarse, tiempo que equivale a 5 ciclos de la oscilación analizada. La Figura 12 muestra el diagrama de bloques del ejemplo y la Figura 13 el resultado de la simulación.

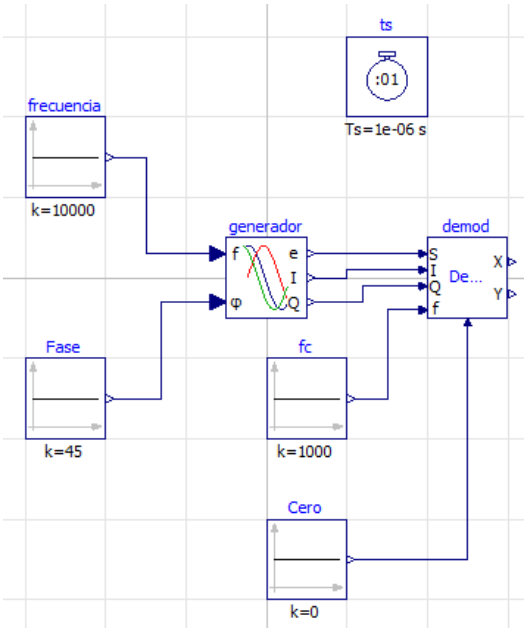



**Figura 12. Ejemplo DemodulaTest**

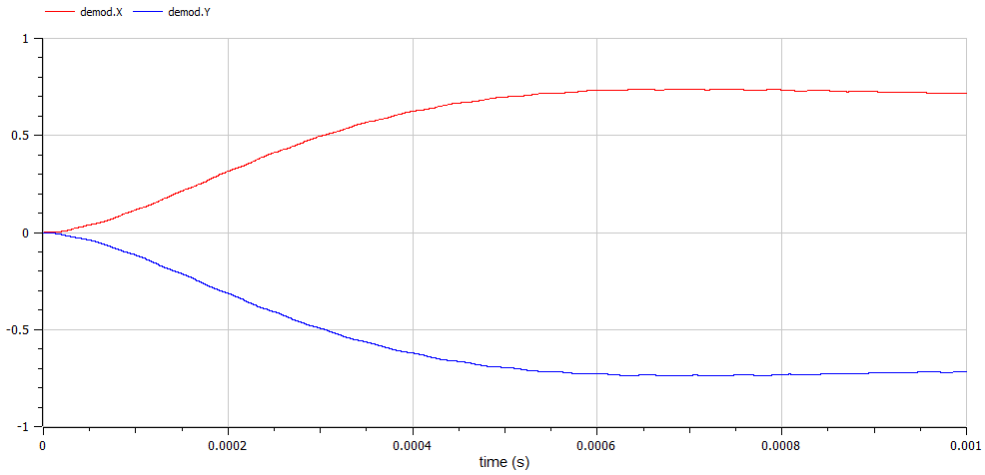



**Figura 13. Señales x e y procedentes del demodulador**

### 4.4.3 PromedioTest

Verifica el correcto funcionamiento del filtro FIR que promedia ciclos con distintas configuraciones (Figura 14). En la primera está deshabilitado, en la segunda promedia un ciclo y en la tercera, dos. Se emplean pocos ciclos de muestreo en cada periodo para que se aprecie mejor el resultado y se eligen las frecuencias de manera que una no sea múltiplo de la otra para que el promedio sea no nulo. En la Figura 15 podemos ver que la salida sin filtrar (en rojo) es una onda sinusoidal muestreada, que el filtro de 1 ciclo (azul) se actualiza cuando la señal sin filtrar tiene un máximo, que equivale a que la referencia usada pase por cero en sentido ascendente y tiene mayor amplitud que la que genera el filtro de 2 ciclos (verde), como era de esperar.

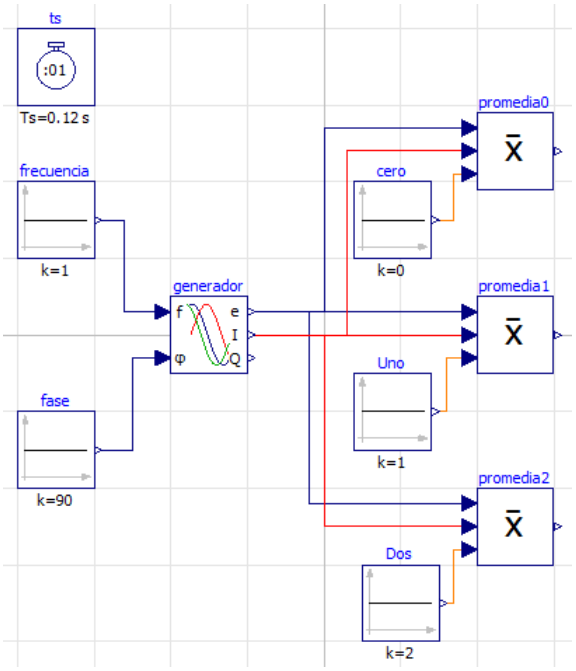



**Figura 14. Ejemplo PromedioTest**

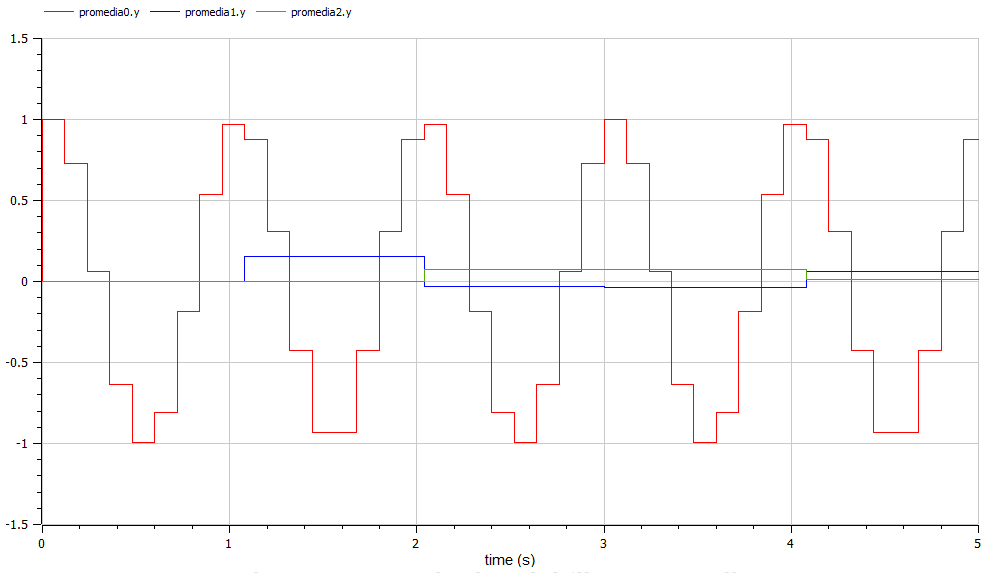



**Figura 15. Resultados del filtro promedio**

### 4.4.4 PolaresTest

Este ejemplo realiza las pruebas unitarias del convertidor de coordenadas cartesianas a polares, generando una señal de amplitud constante y fase variable (Figura 16). Una vez superado el estado transitorio el módulo del vector (variable R) permanece constante (Figura 17) y su fase (variable Theta, Figura 18) varía linealmente, como era de esperar.

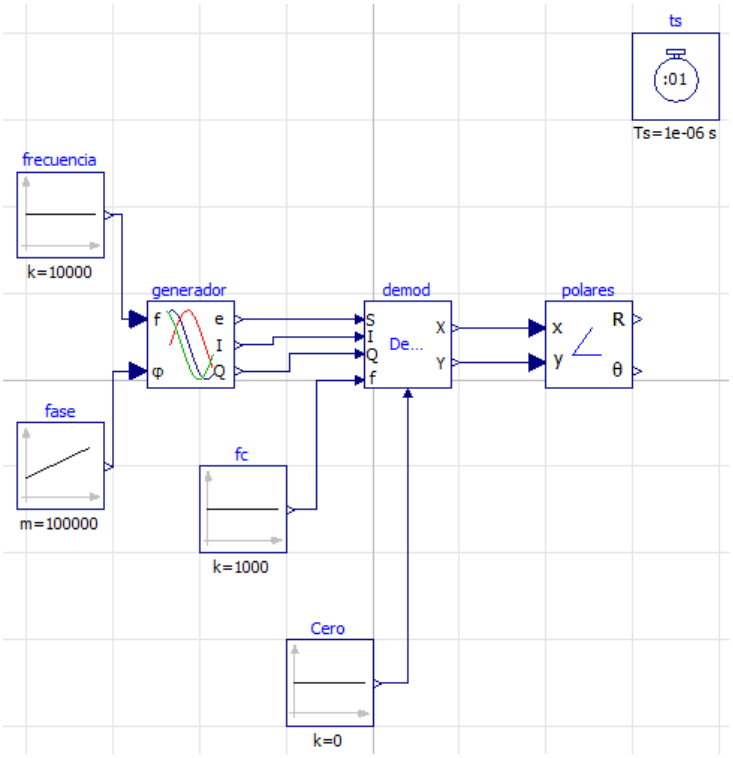



**Figura 16. Ejemplo PolaresTest**

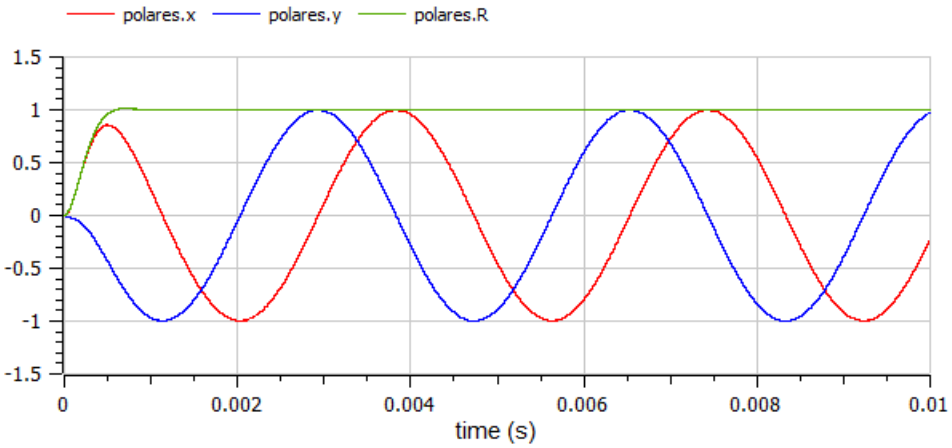



**Figura 17. Coordenadas polares y módulo del resultado**

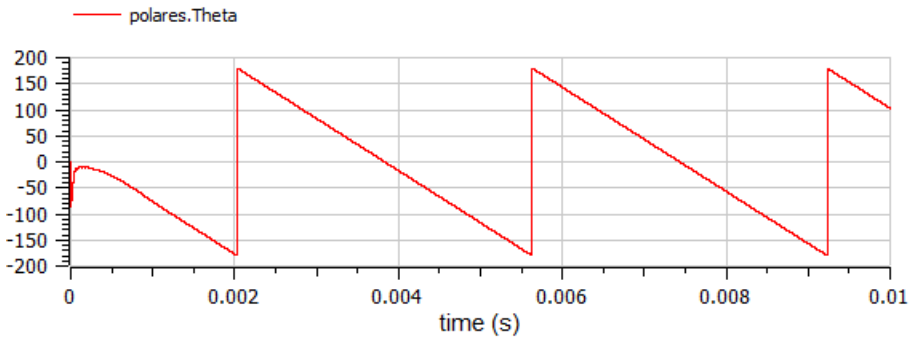



**Figura 18. Fase calculada**

### 4.4.5 LockInBasicoDiodo

Verifica el correcto funcionamiento de la versión básica preconfigurada del amplificador lock-in mediante un experimento que permite medir la resistencia dinámica de un diodo en un determinado punto de polarización (Figura 19). Para ello se polariza el diodo con una tensión constante y sobre ella se suma otra de corriente alterna y poca amplitud, de manera que la respuesta del diodo sea prácticamente lineal. Se mide la corriente que circula por el diodo mediante una resistencia de valor despreciable. Calculando el cociente entre la amplitud de excitación (tensión) y la amplitud de la corriente medida se obtiene la resistencia dinámica en ese punto de polarización. La Figura 20 muestra el resultado de la simulación, donde se aprecia que la amplitud de estabiliza en un $1.8 \cdot 10^{-8}$V, que con la resistencia usada equivale a $1.8 \cdot 10^{-5}$A. La resistencia dinámica del diodo sería:

$$R_D = \frac{10^{-6} V}{1.8 \cdot 10^{-5} A} = 0.056 \Omega$$

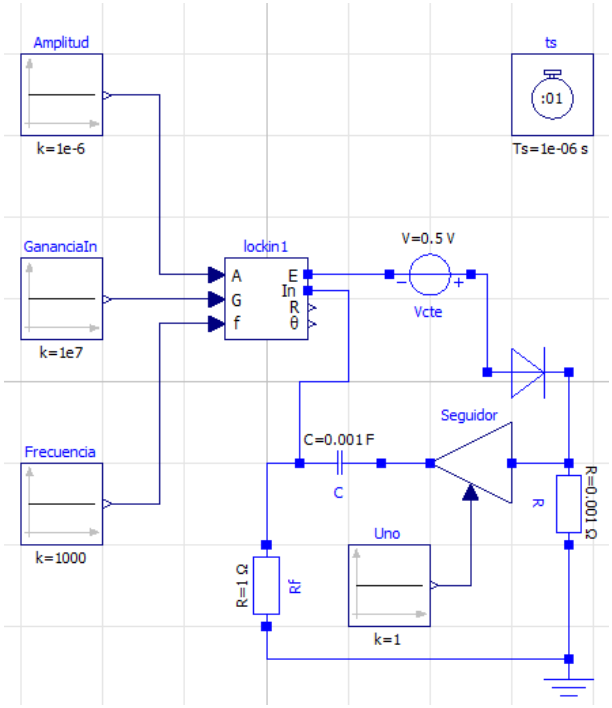



**Figura 19. Ejemplo LockInBasicoDiodo**

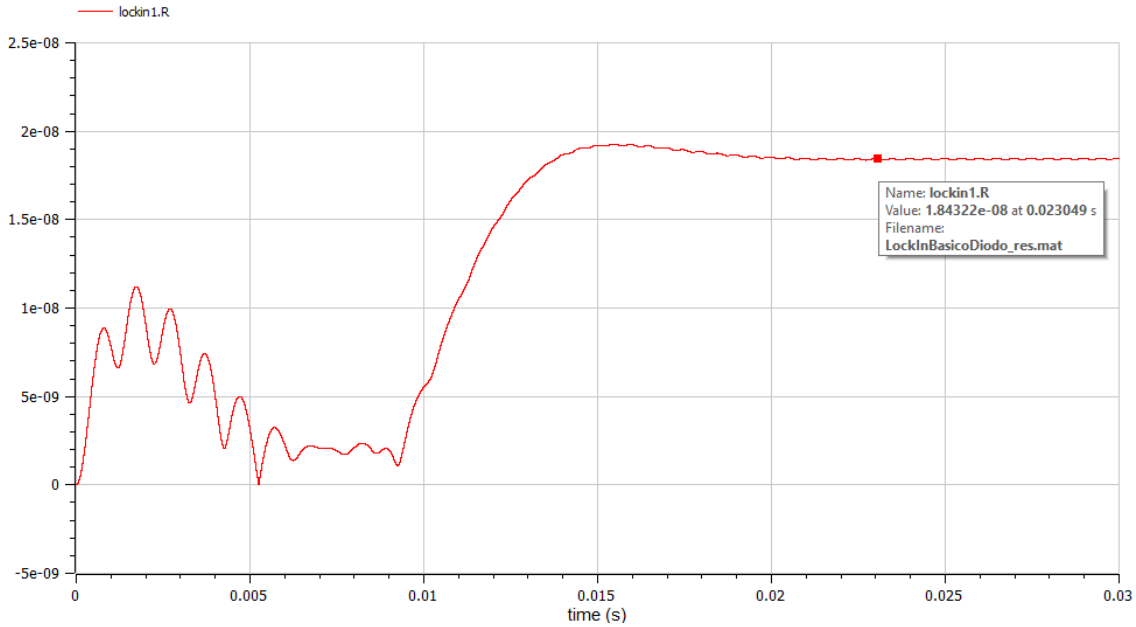



**Figura 20. Amplitud medida (V)**

### 4.4.6 LockInBasicoRLC

Verifica el correcto funcionamiento de la versión básica preconfigurada del amplificador lock-in excitando un sistema resonante (Figura 21). Según los valores de los componentes, la respuesta en voltaje debería tener un máximo en 112540Hz, valor muy cercano al que se obtiene en la simulación (Figura 22). Las oscilaciones posteriores se deben a que el cambio en la frecuencia de excitación es demasiado rápido y no da tiempo a que el sistema se estabilice.

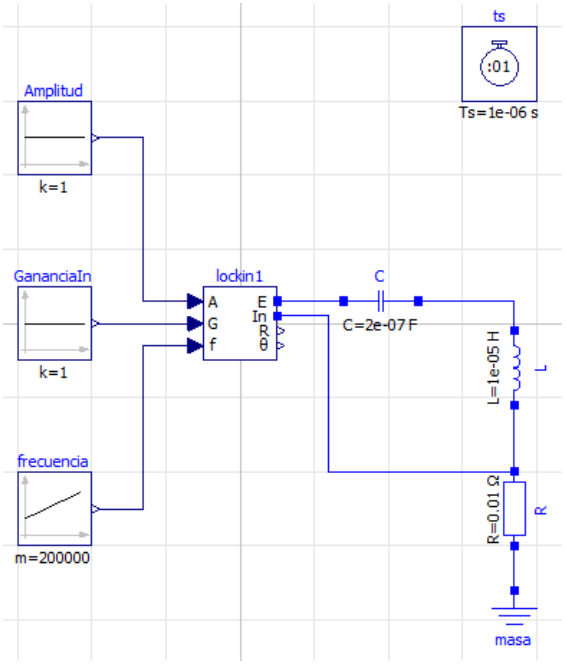



**Figura 21. Ejemplo LockInBasicoRLC**

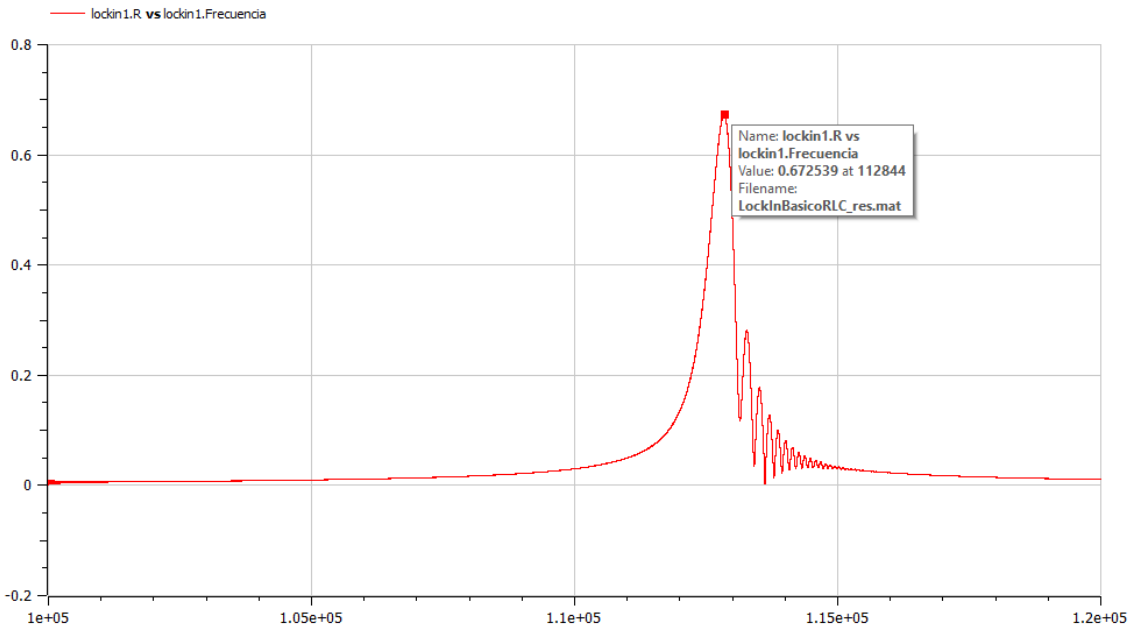



**Figura 22. Amplitud de la oscilación con el máximo en la frecuencia de resonancia**

### 4.4.7 PllRlc

Ejemplo que muestra el correcto funcionamiento del PLL para mantener el sistema en resonancia (Figura 23). Emplea el mismo circuito del ejemplo LockInBasicoRLC, del que se conoce la frecuencia de resonancia teórica y se ha medido la fase que introduce el sistema de medida, que se compensa para que el valor de fase medido en resonancia sea nulo. Activa el amplificador lock-in y tras esperar un tiempo a que se estabilice el

sistema enciende el PLL en t=0.02s, momento en el que empieza a variar la frecuencia de manera automática hasta la de resonancia. El resultado de la simulación se muestra en la Figura 24.

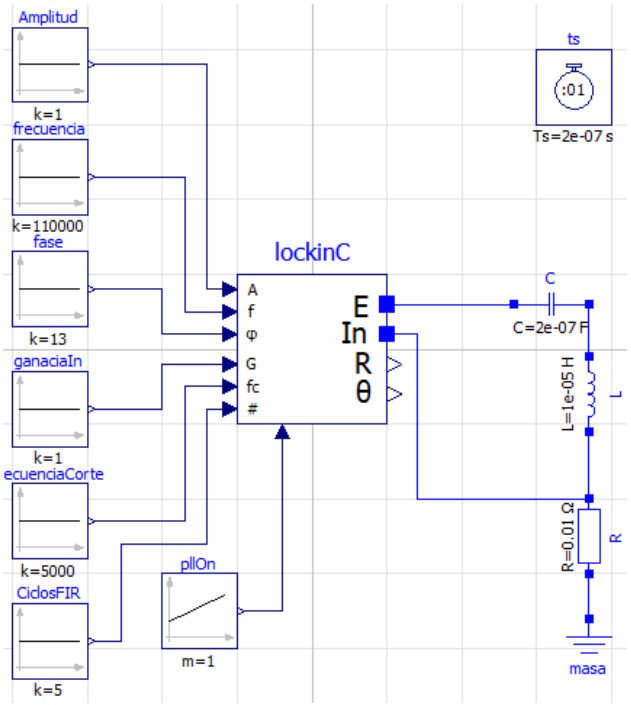



**Figura 23. Ejemplo PllRlc**

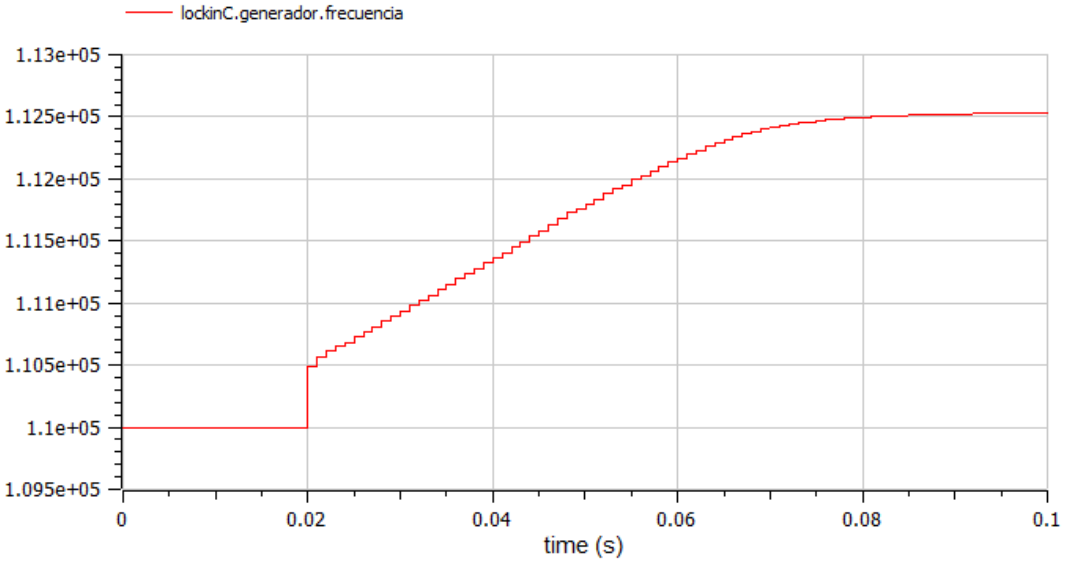



**Figura 24. Simulación de la variación de frecuencia debida al PLL**

### 4.4.8 LockInFiltros

Ejemplo que añade ruido a la simulación, de manera que se pueda comprobar el efecto de los distintos filtros sobre las señales de amplitud y fase (Figura 25). En este

caso no se incluyen capturas de pantalla de los resultados de la simulación porque no se recomienda ninguna configuración de filtros concreta, se deja a criterio del usuario.

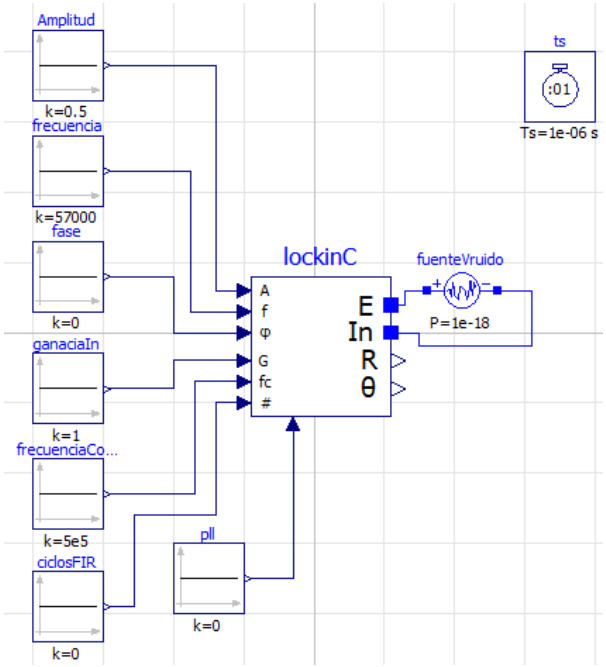



**Figura 25. Ejemplo LockInFiltros**

## 4.5 Conclusiones de las simulaciones

En este capítulo hemos presentado el desarrollo y validación de una librería para poder simular amplificadores lock-in digitales, sistemas que debido a su buen rechazo al ruido son muy empleados para caracterizar dispositivos o como parte de controladores de instrumentación científica. La librería Lock-in incluye los módulos necesarios para formar distintas configuraciones, así como dos versiones de distinta complejidad listas para usar y varios ejemplos que se han empleado para validar los modelos.

Aparte del uso obvio para los ingenieros que deban desarrollar estos sistemas, como es el tema de este TFM, la librería se puede emplear con fines didácticos, de manera que quien empiece a trabajar en esta técnica pueda realizar simulaciones sencillas y analizar las distintas señales y el comportamiento de los distintos elementos que forman el sistema completo.

# CAPÍTULO 5. IMPLEMENTACIÓN

Este capítulo se puede considerar la parte central del TFM y en él vamos a tratar la implementación concreta del amplificador lock-in basado en un microcontrolador, desde la elección de componentes al diseño e implementación del *hardware* y el *software* asociado.

## 5.1 Especificaciones

Lo ideal en un proyecto de ingeniería sería empezar con unas especificaciones bien definidas y realizar una selección de componentes en base a esas especificaciones, pero aquí no ha sido exactamente ese el caso. Y no lo ha sido por varios motivos, el principal es que en el proyecto pretendemos estudiar el límite al que se puede llegar empleando un microcontrolador como DSP en un amplificador lock-in y para eso lo primero que tenemos que hacer es analizar las capacidades de los microcontroladores actuales y luego estimar unas especificaciones a las que creamos que podemos llegar. Sí que hay unos objetivos mínimos que se deben satisfacer para que el proyecto tenga sentido, que se pueden resumir en que el amplificador lock-in resultante se pueda emplear para usos poco exigentes, que podría ser analizar señales de hasta 1kHz con una precisión del 1% una vez calibrado. También disponemos de unas estimaciones groseras de las especificaciones a partir de trabajos previos [14], que indican que con un microcontrolador de 240MHz de frecuencia de reloj y un sintetizador externo consiguen manejar ADC de bus paralelo de 2MSPS, que será el orden de magnitud de la frecuencia de ciclo objetivo.

Una de las características deseables de cualquier instrumento electrónico es que tenga poco ruido. Existen múltiples maneras de caracterizar el ruido, entre ellas las curvas de densidad espectral de ruido, la relación de rechazo al ruido (*noise rejection ratio*), la distorsión harmónica total cuando se trabaja con señales sinusoidales, etc. En los amplificadores lock-in estos valores se refieren a la señal a analizar y se miden en las señales de resultados, típicamente en la amplitud medida (R). Debe tenerse en cuenta que esta amplitud medida es el resultado de filtrar ciertos cálculos intermedios, por lo que los resultados dependerán de la frecuencia de corte y el orden de este filtro. En la Tabla 2-1 vimos que el que probablemente sea el fabricante de más prestigio a nivel mundial de este tipo de instrumentos expresaba el ruido empleando una frecuencia de corte de 1Hz en la salida, que será lo mismo que hagamos nosotros. El dato que mostraban era la densidad espectral de ruido, cuya distribución es de esperar que dependa de la frecuencia de trabajo, ya que el primer cálculo que realiza el amplificador lock-in es desplazar en frecuencias la señal de entrada. Caracterizar el ruido de esta manera puede ser una tarea ardua y que requiera equipamiento especializado. Lo que es más sencillo de medir es la relación de rechazo al ruido, es decir, qué fracción de ruido de una determinada frecuencia aparece en frecuencias cercanas a ellas, valor que se suele expresar en dB. A modo de ejemplo, una relación de 40dB significaría que veríamos un ruido del 1% de la amplitud original, algo justo pero suficiente para validar el resto del diseño, y 60dB equivaldría al 1‰, un valor ya adecuado. En este caso un ruido de 1V aparecería en los resultados como 1mV.

Otra de las características, quizá de las menos normalizadas entre distintos gremios usuarios de aparatos electrónicos sea los límites de tensión de trabajo. Hace años era habitual que la instrumentación científica empleara señales analógicas en el rango de +/-10V. Con el desarrollo de la electrónica de bajo consumo las tensiones en general han disminuido, no solo en los buses digitales sino también en señales analógicas. Es común encontrar amplificadores lock-in comerciales en los que las interfaces analógicas tienen un rango de ±10V pero también de ±1V, ±2V, ±5V o incluso varias de estas opciones seleccionables por el usuario. Para el objetivo de este trabajo no es una característica que nos deba importar en exceso, pero diseñaremos en base a los valores que se solían emplear antaño y aún se mantienen en muchos instrumentos, es decir, ±10V. En cualquier caso, tendremos en cuenta que para un gran número de aplicaciones el rango necesario será menor, por lo que no nos preocuparemos en exceso por cumplir esta característica, siendo suficiente con llegar a ±1V si eso simplificara el diseño. Podríamos definir ya dos conjuntos de especificaciones un poco más formales, las mínimas para considerar que el proyecto es útil en alguna aplicación y otras más ambiciosas que serían el objetivo que perseguir y significarían que realmente puede ser un producto competitivo incluso comercialmente, que sería el objetivo deseado.

**Tabla 5-1. Resumen de especificaciones**

| | **Básico** | **Ideal** |
|---|---|---|
| **Frecuencia de muestreo (kHz)** | 10 | 5000 |
| **Ancho de banda (kHz)** | 1 | 500 |
| **Rechazo al ruido (dB)** | 40 | 60 |
| **Rango analógico (±V)** | 1 | 10 |
| **Resultados** | X, Y analógico (1kHz) | X, Y, R, θ, analógico (10kHz) y digital (USB/ethernet) |
| **Configuración** | Fija | USB/Ethernet |

## 5.2 Selección de componentes

En los últimos años se han popularizado las fábricas de placas de circuito impreso, ofreciendo precios de fabricación y plazos de entrega que hacen cuestionarse la utilidad de otras herramientas de prototipado, como la fabricación casera con productos químicos. Algunas de estas empresas ofrecen también el servicio de montaje e incluso disponen de su propio almacén de componentes electrónicos, de manera que a partir de los ficheros de fabricación son capaces de fabricar la placa y ensamblar los componentes, facilitando enormemente el proceso de desarrollo a quienes no tienen mucha habilidad con las herramientas de soldadura o simplemente carecen de los medios adecuados, y todo ello a unos precios realmente bajos. Hemos optado por esta opción para la mayoría de los componentes de montaje superficial, solo ha habido uno que hemos tenido que soldar a mano porque la fábrica no disponía de la referencia concreta ni ninguna de características parecidas. Los conectores y el resto de los componentes de inserción sí se han soldado manualmente de la manera tradicional. Este método de fabricación en base al inventario concreto de una empresa de montaje ha condicionado la elección de componentes, buscando en este primer prototipo la

validación de la idea empleando referencias comunes y disponibles, más allá de las mejores características de un catálogo más amplio, lo que encarecería sustancialmente los costes de fabricación. Hemos usado los servicios de la empresa JLCPCB.

### 5.2.1 Microcontrolador

Sin duda es el elemento más importante y el que requiere más estudio previo, ya que será el que determine la capacidad de cálculo, los elementos que podrá controlar según sus entradas y salidas, los buses que tenga implementados en *hardware*, los niveles de tensión que emplee, los tiempos de acceso a los puertos, etc. Este último punto es fundamental, ya que existen microcontroladores con gran potencia de cálculo que en cambio tienen tiempos de acceso a los periféricos muy elevados, aunque solo sea a las entradas y salidas digitales de propósito general (GPIO, *General Purpose Input/Output*). En cada ciclo de muestreo deberemos actualizar un DAC y leer un ADC y es de esperar que cada una de estas operaciones requiera varios accesos a GPIO si emplean un bus paralelo o al menos un acceso a un bus serie tipo $I^2C$ (*Inter-Integrated Circuit*) o SPI (*Serial Peripheral Interface*). Lamentablemente la información relativa a los tiempos de acceso reales a los periféricos no la suelen proporcionar los fabricantes o si lo hacen no suele ser la que destaquen en sus especificaciones principales y haya que buscarla en los manuales de referencia, donde puede venir como un comentario más dentro de algún apartado que no llame mucho la atención. En microcontroladores modernos, dada su complejidad, estos manuales suelen contener cientos de páginas.

Otra de las características deseables del microcontrolador para poder llevar a término el proyecto en plazos y costes adecuados a lo que es un TFM es que sea fácil empezar a trabajar con él, de manera que no gastemos todo el tiempo del proyecto en ponerlo en marcha. Hablar de microcontroladores en las últimas décadas, y en concreto de facilidad de uso, nos lleva al ecosistema Arduino, no solo las placas propias de la marca sino a todas las otras compatibles en mayor o menor medida con las herramientas disponibles para estos sistemas. Los inconvenientes de esta opción son principalmente dos:

1. La potencia de las placas de Arduino en general es muy limitada. La idea para la que fueron diseñadas no es realizar tareas complejas y potentes, sino acercar el mundo de los microcontroladores y el bricolaje digital al público en general. Una excepción fue su modelo Arduino Due, que contaba con gran cantidad de pines de entrada y salida configurables, varios periféricos con implementaciones *hardware* que controlaban distintos buses con múltiples unidades de cada uno, etc. Este modelo ya tiene 10 años y no se puede considerar puntero, aunque su CPU (SAM3X) de 32 bits de 84MHz podría ser suficiente para una versión básica del amplificador si no hubiera ninguna alternativa mejor. Afortunadamente, recientemente Arduino ha decidido expandir su mercado a otros sectores, como el industrial, con placas que montan micros de mayor potencia y un nuevo estándar en sus conectores, que ofrece 160 pines, muchos de ellos disponibles

para el usuario. La primera (y a la fecha de escritura de esta memoria, única[2]) placa de esta gama se llama Portenta y monta un microcontrolador de ST Microelectronics (STM32H747) con dos núcleos asimétricos, un Cortex® M7 de hasta 480MHz y un Cortex® M4 de hasta 240MHz (Figura 26), que indica que se pueden comunicar mediante la técnica RPC (*Remote Procedure Call*). Aparte de las diferencias en la velocidad de reloj, el núcleo M7 ofrece un set de instrucciones mayor e implementación *hardware* de muchas unidades, como una unidad en coma flotante de doble precisión, paralelización de tareas, ejecución preventiva, latencia de 0 ciclos en bucles, etc. Existe una versión de esta placa llamada Portenta Lite, que monta el mismo chip pero carece de algunas características de seguridad y comunicaciones, que podría ser adecuada para este proyecto, y cuyo precio ronda los 60€.

2. La sencillez de uso suele estar reñida con la eficiencia, y Arduino no es una excepción. Si la misma instrucción que enciende o apaga un LED funciona en decenas de placas con microcontroladores de distintos fabricantes, esto es porque la función que empleamos lo primero que hace es determinar qué placa es, comprobar si los argumentos son correctos para esa placa, por ejemplo, que el número de pin sea menor del total de pines del modelo que estemos usando, etc. Y todo esto se hace en tiempo de ejecución, lo que provoca que la eficiencia sea menor que si algunas de las tareas se realizaran en tiempo de compilación. Sirva de ejemplo indicar que el lenguaje de Arduino no proporciona ninguna manera de acceder a varios pines a la vez, en su lugar hay que acceder a los registros específicos de cada controlador, lo que hace que el código deje de ser portable, pero es mucho más eficiente. Otras tareas, como modificar el mapa de memoria del controlador o indicar qué partes de código o qué variables deben ubicarse en qué zonas, aunque son posibles, no son fáciles. Por tanto, la ventaja de usar un entorno tan sencillo como Arduino o incluso Visual Studio Code, que es más potente, pierden parte de su sentido cuando queremos llegar al límite que nos permita el *hardware*, porque vamos a tener que saltarnos muchas de las facilidades que nos ofrecen estos entornos.

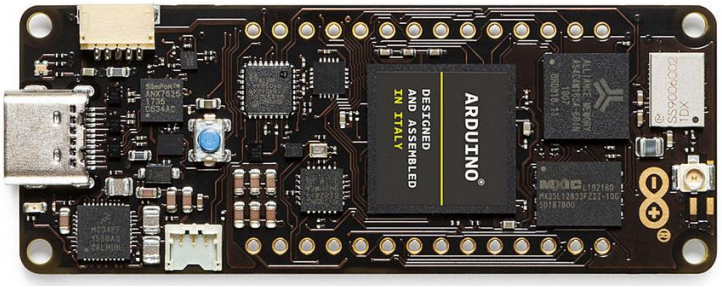



**Figura 26. Arduino Portenta**

Lo que sí creemos que es fundamental es disponer del microcontrolador que vayamos a usar en un módulo con los componentes necesarios para un prototipado

[2] En realidad, existen dos versiones, la normal y una denominada “Lite”, pero la única diferencia detectada es que una de ellas no monta algunos componentes opcionales que la otra sí lleva.

rápido, ya se llame módulo de evaluación, placa de pruebas, placa de microcontrolador o cualquier otro nombre similar, de manera que podamos reducir el tiempo de desarrollo. Afortunadamente la mayoría de los fabricantes ofrecen este tipo de placas a precios económicos.

Tenemos por tanto ya tres candidatos como microcontroladores o sus módulos, el PIC32MZ que ya está validado y ofrece buenos resultados, el Arduino Portenta con su STM32H747 y si ninguno de ellos diera buen resultado incluso el ya veterano Arduino Due. El doble núcleo del Portenta puede ser una ventaja, ya que podemos separar distintas tareas entre ellos, algunas con requisitos de tiempo real críticos y otras, como la comunicación con el usuario, sin tantas restricciones. En los últimos años también ha sido muy popular el modelo ESP32, con un precio realmente rompedor (se puede conseguir por menos de 5€ en tiendas online) y unas especificaciones muy completas. Aparte de la conectividad inalámbrica, que en este proyecto no es importante, ofrece dos núcleos de 32 bits de 240MHz, por lo que puede ser el cuarto candidato que analizar.

Volviendo al tema de los tiempos de acceso a periféricos y las tareas en tiempo real estricto o duro (*hard real time*), la principal será leer el ADC y actualizar el DAC. Habrá otras, como controlar las etapas analógicas, actualizar resultados o comunicarse con el usuario, que no requieran tasas de refresco tan altas. Si buscamos frecuencias de conversión del orden de los MSPS nos encontramos principalmente con dos alternativas en cuanto a interfaces de conexión: SPI y paralelo. El bus SPI típicamente trabaja a frecuencias máximas de 50 MHz, en muchos casos solo llega hasta 30MHz. Suponiendo que el convertidor necesite 16 ciclos para actualizarse y uno adicional para la señal de activación de chip podríamos llegar a una tasa teórica de unos 3MSPS, que estaría en el rango aceptable. El problema es que estos tiempos teóricos muchas veces son difíciles de cumplir y están condicionados por los recursos que ofrezca el controlador y su entorno de programación. Es común que, aunque se puedan enviar datos a esas tasas, cada comunicación tarde en iniciarse centenas de ns o incluso varios µs, y una vez más, esa información no suele ser fácil de conseguir. Si el µControlador permite el acceso directo a memoria (DMA, *Direct Memory Access*) desde el periférico quizá sí se puedan conseguir esas tasas de refresco y además la carga de la CPU sería mínima, limitándose a la sincronización para determinar cuándo está lista una nueva lectura del ADC. En cualquier caso, parece más adecuada la comunicación paralela, en la que podemos acceder directamente a la información de los pines, por sencillez y por velocidad de acceso. Si los tiempos de acceso son suficientemente pequeños y reproducibles y el ancho del puerto de entrada/salida tiene los suficientes bits, simplifica mucho la sincronización y permite leer el ADC y actualizar el DAC con una sola instrucción. Una vez más, en teoría, ya que puede haber señales de selección de chip, de reloj, etc., que hagan que se necesiten varios ciclos. Estas señales suelen estar pensadas para usar buses compartidos, de manera que el ADC escriba los datos solo cuando se le requiera o el DAC lea el bus solo cuando los datos que ha puesto el microcontrolador sean los suyos. Según el componente concreto puede existir la opción de funcionamiento en modo continuo, aunque quizá entonces la sincronización sea más complicada. Podemos estimar, en una primera aproximación, que cada componente, DAC y ADC, necesitará dos accesos a señales: uno de control y otro de datos. El de control del ADC puede ser

simplemente activar una señal de disparo para iniciar la conversión y el del DAC activar su señal de actualización una vez el dato esté cargado en el bus paralelo, para evitar picos debidos a distintos retardos en la propagación de las señales. Quizá se puedan unir varios de los accesos, como las señales de control de ambos elementos, quedando en 3 ciclos totales para los dos dispositivos.

Para realizar las pruebas de tiempos de acceso que decíamos que solían venir poco documentadas hemos empezado por microcontroladores que teníamos de proyectos previos o de alta disponibilidad en tiendas y coste reducido. Recordemos la situación actual del mercado de los semiconductores, en que las unidades disponibles de algunos dispositivos son muy limitadas o nulas. Nuestra primera prueba fue con el ESP32, que considerábamos un buen candidato al tener bastantes señales de entrada y salida de usuario y dos núcleos de 240MHz. Obtuvimos unos tiempos de acceso de escritura a los pines de 50ns, que si suponemos 3 ciclos en total darían 150ns, algo más de 6MSPS, mejorando los valores teóricos del bus SPI incluso con aceleraciones *hardware* tipo DMA. Estos 50ns son 12 ciclos de reloj. Durante estas pruebas aparecieron dos aspectos que hay que tener en cuenta en general cuando se trabaja con microcontroladores y en particular cuando estos son modernos. Aunque hace mucho tiempo que existen las interrupciones, que pausan la ejecución normal del programa para atender rutinas de mayor prioridad, dada la complejidad de los sistemas actuales, esta tecnología puede provocar algunos problemas de cierta dificultad en su solución. Por una parte, si queremos manejar los convertidores en tiempo real duro con tiempos de ciclo por debajo del microsegundo, no interesa que el sistema atienda otras rutinas. Para esto es especialmente bueno tener dos núcleos, se podrían deshabilitar las interrupciones en uno de ellos y en el otro realizar esas tareas periódicas necesarias para que el sistema siga funcionando con normalidad. Uno de los problemas asociados a esta solución es el sistema de detección de cuelgues del microcontrolador, normalmente conocido como perro guardián o por su término en inglés *Watch Dog Timer* (WTD). Este mecanismo consiste en un temporizador que controla el tiempo que hace desde la última vez que el código indicó que todo iba bien mediante una instrucción especial. Podemos incluir una llamada a esta instrucción en el código que maneja los convertidores o podemos deshabilitar el WDT. La segunda opción es más eficiente, aunque perdemos la posibilidad de detectar este tipo de fallos. El problema es que tanto en el entorno de Arduino como en Visual Studio Code, esta opción viene activada por defecto y no es fácil cambiarla. La solución es usar las herramientas propias del fabricante, compilando en línea de comandos y deshabilitando el WDT desde su propia herramienta (Figura 27), lo que complica el desarrollo. Estos problemas no son motivo suficiente para descartar la opción, pero si nos hacen tenerlos en cuenta frente a otras alternativas en las que el sea más sencillo.

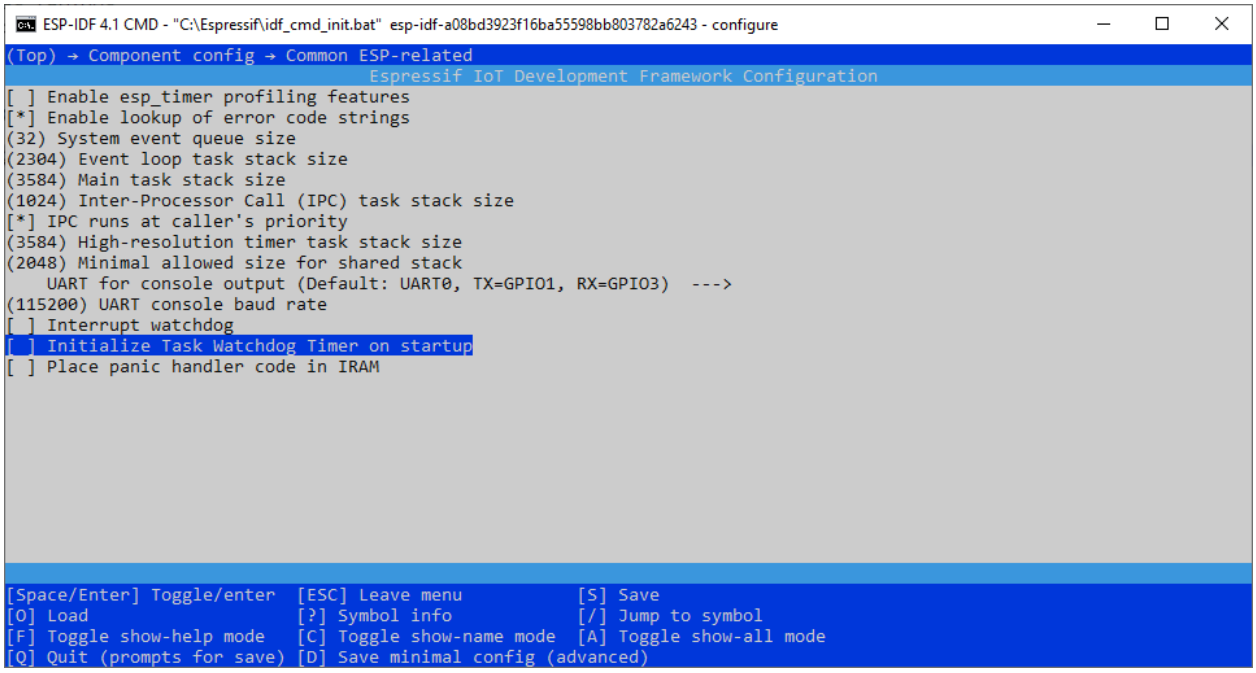



**Figura 27. Selección del WDT en ESP32**

La siguiente prueba fue con un Arduino Due accediendo directamente a los registros, es decir, saltándonos la función digitalWrite(). El acceso así es mucho más rápido, solo dos ciclos, que equivale a unos 24ns. Como referencia, cada acceso usando la función de alto nivel tarda 2.24µs, unas 100 veces más, y solo permite actualizar un bit, en lugar de la palabra completa que actualiza el acceso a registros. Estos resultados son coherentes con las pruebas que han hecho otros usuarios. La Tabla 5-2 reproduce el resultado de esas pruebas en las que realizaron 100 ciclos de escritura en distintos microcontroladores obteniendo el tiempo total y el número de ciclos de reloj de cada una de estas actualizaciones [22].

**Tabla 5-2. Ciclos de acceso de escritura de GPIO de distintos controladores**

| | F072 | F103 | L476 | F446 | SAM3X | SAMD21 | UNO |
|---|---|---|---|---|---|---|---|
| **Reloj (MHz)** | 48 | 64 | 80 | 180 | 84 | 48 | 16 |
| **Tiempo total (µs)** | 4.33 | 3.29 | 1.33 | 0.58 | 2.38 | 8.96 | 12.63 |
| **Número de ciclos** | 2 | 2 | 1 | 1 | 2 | 4 | 2 |

De la tabla anterior deducimos que los 12 ciclos de acceso del ESP32 son un valor muy por encima de la media, por lo que puede ser interesante probar con algún otro modelo, de arquitectura más parecida a los recogidos en la tabla, pero más moderno que el Due. Las primeras 4 columnas son modelos de ST Microelectronics [23], fabricante del chip que monta el Arduino Portenta, por lo que parecía obligatorio probar esta placa o alguna similar. Posee gran potencia de cálculo, mucha disponibilidad y es relativamente barata, es de suponer que la puesta en marcha para una primera evaluación sea rápida y existen distintos accesorios como placas de expansión, que facilitarían el desarrollo. Las pruebas de los tiempos de respuesta con este controlador mostraron unos resultados algo distintos de los esperados. Fueron muy básicas, accediendo directamente a los

registros de los GPIO de manera que cambiase el valor lógico en cada instrucción, y medimos la frecuencia de la señal de salida en un osciloscopio, siendo el tiempo de ciclo el doble del de actualización, ya que en cada ciclo la señal se pone a nivel alto y bajo. Recordemos que disponemos de dos núcleos, que según las especificaciones funcionan uno a 480MHz (M7) y el otro a 240MHz (M4). La manera de asignar el código a cada núcleo usando las librerías de Arduino es realmente sencilla, basta con emplear uno u otro *#define* y cargar el código en el núcleo adecuado. En este sentido era lo esperado, la facilidad de uso que ha ayudado a Arduino a ser lo que es hoy en día. Midiendo el periodo en el núcleo M7 obtuvimos 60 ns, lo que serían 30 ns por actualización, que supondría 14.4 ciclos de reloj por acceso. Evidentemente el número debe ser entero, quizá la precisión de la medida no fuera la adecuada, pero al menos tenemos un orden de magnitud, mucho mayor de lo esperado, casi comparable al ESP32, pero con un precio 10 veces superior. Al realizar la misma prueba con el núcleo más básico y en teoría más lento, el tiempo de ciclo fue 10ns, es decir, 5 ns por acceso. Una vez más, no tiene sentido para una frecuencia de 240MHz, ya que serían 1.2 ciclos. La realidad es que es un solo ciclo, pero de 200MHz. Por alguna razón, y sucede lo mismo en los ejemplos del fabricante del controlador, el reloj está configurado para funcionar a 200MHz en lugar de 240MHz y el del núcleo M7 a 400MHz en lugar de 480MHz, desperdiciando en ambos casos un 17% de la potencia de cálculo. La frecuencia de 400MHz en el núcleo M7 significa que en realidad emplea 12 ciclos en actualizar el GPIO. Dejando de lado este hecho, el dato de un único ciclo en el núcleo M4 es el mejor que se podría obtener y mejora mucho los tiempos de sus competidores. La diferencia entre un núcleo y el otro, y el hecho de que tarde menos en el M4 se explican por la arquitectura interna del microcontrolador, que tiene 3 dominios, según podemos consultar la nota de aplicación AN5557 del fabricante donde encontramos el diagrama de conexiones que reproducimos en la Figura 28. En ella observamos que cada núcleo tiene asociados una serie de componentes a los que se conecta directamente y que existen buses para acceder a los recursos de otro dominio.

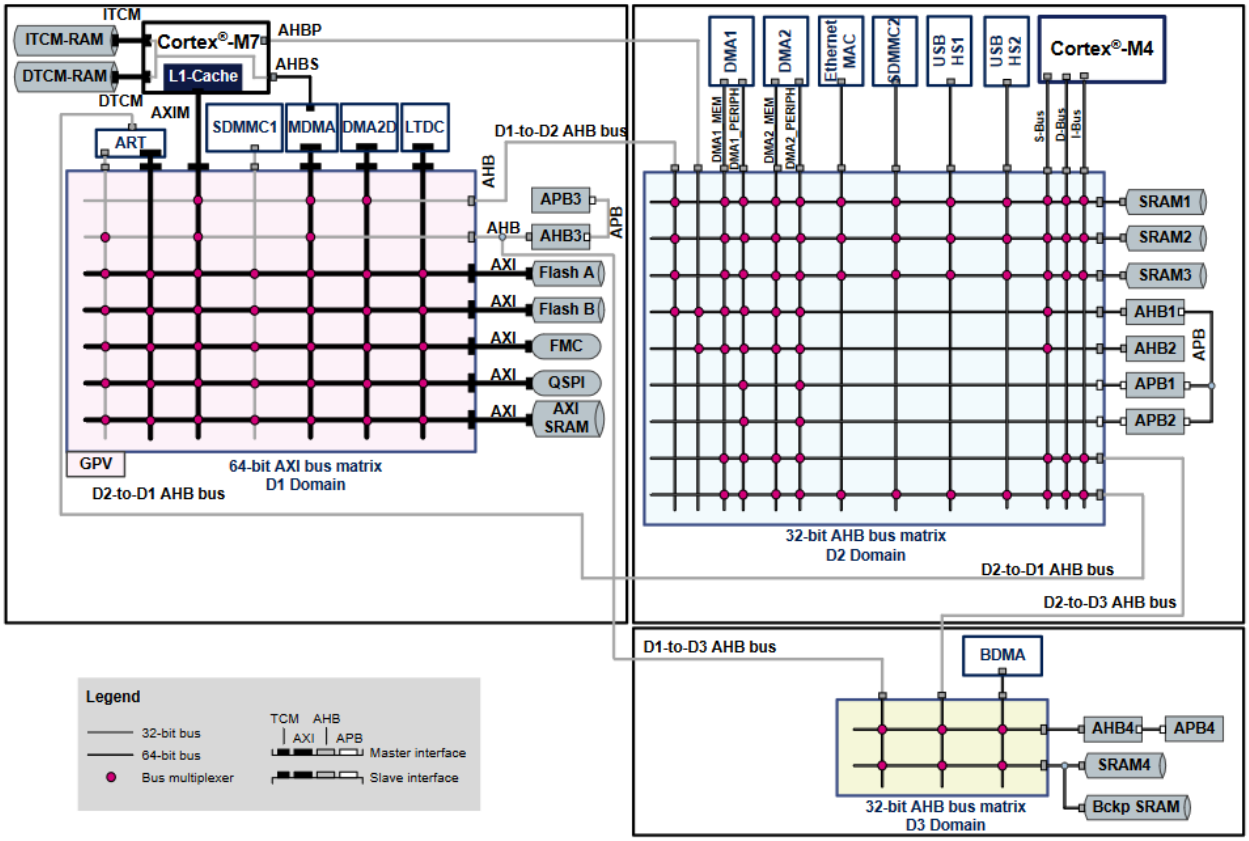



**Figura 28. Esquema de conexiones de los modelos STM32H7**

Suponiendo un total de 3 ciclos del núcleo M4 configurado a 200MHz para manejar un DAC y un ADC de bus paralelo serían solo 15ns, es decir, 67MHz. Por supuesto, en cuanto añadamos lógica los tiempos aumentarán y quizá el acceso de lectura sea más lento que el de escritura, pero por el momento es la mejor opción. Midiendo estos tiempos de lectura obteneos 18 ciclos desde el núcleo M7 y 1 ciclo desde el M4. La conclusión lógica es que para manejar los convertidores es mejor emplear el núcleo M4 y para realizar los cálculos el M7, más rápido y con más funcionalidades implementadas en *hardware*. El modo de comunicación basado en llamadas a procedimientos remotos en general no es el más adecuado para sistemas de baja latencia, como debe ser este, por lo que habrá que buscar una alternativa o distribuir con especial cuidado las tareas entre los dos núcleos.

A la vista de los buenos resultados obtenidos con el Arduino Portenta decidimos buscar en el catálogo del fabricante del chip otros controladores de características similares o superiores, llegando a la Tabla 5-3 [24], que muestra una comparativa de los modelos disponibles en la familia H7, la de más alta gama. Hay gran variedad de modelos, pero dos de las filas destacan entre las demás, por tener dos núcleos y frecuencias de trabajo elevadas, el STM32H744/757, que monta el Arduino, y el STM32H7455/755. Dentro de cada una de estas entradas, la diferencia entre modelos se debe a la presencia o ausencia de módulos *hardware* para aplicaciones criptográficas, algo que por el momento no nos interesa.

**Tabla 5-3. Comparativa de microcontroladores de gama alta de ST Microelectronics**

## STM32H7 MCU Series 32-bit Arm® Cortex®-M7 or Cortex®-M7 + Cortex®-M4

**CORE, MEMORIES AND ACCELERATION**
- Single-core Cortex-M7 up to 550 MHz
- Dual-core Cortex-M7 480 MHz and Cortex-M4 240 MHz
- Flash and RAM acceleration
- SP-FPU and DP-FPU
- 4 x DMA
- Mathematics (only H723/733/725/735/730)

**CONNECTIVITY**
- Up to 2 x USB2.0 OTG FS/HS
- 2 x SDMMC
- USART, UART, SPI, I²C
- Up to 3 x CAN (2 x FD and 1 x TT)
- HDMI-CEC
- FMC, Dual-mode Quad-SPI or 2 x Octo-SPI
- Camera I/F

**AUDIO**
- 3 x I²S + audio PLL
- 4 x SAI
- 2 x 12-bit DAC
- SPDIF-RX

**GRAPHIC**
- Chrom-ART Accelerator™

**OTHER**
- Crypto/Hash option (except H742)[1]
- Security services option (except H742)
- TRNG
- DFSDM
- 16- and 32-bit timers
- HRTimer (except STM32H7A/H7B/H7B0/H723/H725/H730/H733/H735)
- Up to 3 x 16-bit ADC (up to 3.6 MSPS)
- Analog (compt,AOP)
- Voltage range 1.62 to 3.6 V (except 100-pin and VFQFPN68 packages : 1.71 to 3.6 V)
- Multi-power domains
- -40°C up to 105°C ambiant
- -40°C up to 125°C ambiant[2]

| Product line | $F_{CPU}$ (MHz) | Dual-Bank Flash memory (bytes) | RAM (bytes) | OctoSPI & OTFDEC[3] | Ethernet | Graphic | Power supply | Stop mode (typical) / RAM retention |
|---|---|---|---|---|---|---|---|---|
| **Dual-core lines** | | | | | | | | |
| STM32H747/757[1] | 480 + 240 | Up to 2 Mbytes | 1 Mbyte (incl.128 Kbytes DTCM + 64 Kbytes ITCM + 64 Kbytes backup[1]) + 4 Kbytes backup[2] | | • | TFT-LCD JPEG codec MIPI-DSI | SMPS + LDO | 360 µA / 1MB 250 µA / 768KB |
| STM32H745/755[1] | 480 + 240 | Up to 2 Mbytes | 1 Mbyte (incl.128 Kbytes DTCM + 64 Kbytes ITCM + 64 Kbytes backup[1]) + 4 Kbytes backup[2] | | • | TFT-LCD JPEG codec | SMPS + LDO | 360 µA / 1MB 250 µA / 768KB |
| **Single-core lines** | | | | | | | | |
| STM32H7A3/7B3[1] | 280 | Up to 2 Mbytes | 1,4MB (incl.128K DTCM, 64K ITCM, 1184K+SRAM, 4K backup) | • | | TFT-LCD JPEG codec Chrom-GRC | SMPS + LDO | 32 µA / 1.4MB 28 µA / 32KB |
| STM32H743/753[1] | 480 | Up to 2 Mbytes | 1 Mbyte (incl.128 Kbytes DTCM + 64 Kbytes ITCM + 64 Kbytes backup[1]) + 4 Kbytes backup[2] | | • | TFT-LCD JPEG codec | LDO | 1270 µA / 1MB 910 µA / 768KB |
| STM32H742 | 480 | Up to 2 Mbytes | 692 Kbytes (incl.128 Kbytes DTCM + 64 Kbytes ITCM + 16 Kbytes backup[1]) + 4 Kbytes backup[2] | | • | | LDO | 1270 µA / 692KB 910 µA / 704KB |
| STM32H725/735[3] | 550 | Up to 1 Mbyte | 564KB (incl.128K DTCM, 432KB Syst + 4K bckup) | • | • | TFT-LCD | SMPS[5] + LDO | 200 µA / 564KB |
| STM32H723/733[3] | 550 | Up to 1 Mbyte | 564KB (incl.128K DTCM, 432KB Syst + 4K bckup) | • | • | TFT-LCD | LDO | 520 µA / 564KB |
| **Value line** | | | | | | | | |
| STM32H7B0 | 280 | 128 Kbytes | 1,4MB (incl.128K DTCM, 64K ITCM, 1184K+SRAM, 4K backup) | • | | TFT-LCD JPEG codec Chrom-GRC | SMPS + LDO | 32 µA / 1.4MB 28 µA / 32KB |
| STM32H750 | 480 | 128 Kbytes | 1 Mbyte (incl.128 Kbytes DTCM + 64 Kbytes ITCM + 64 Kbytes backup[1]) + 4 Kbytes backup[2] | | • | TFT-LCD JPEG codec | LDO | 1270 µA / 1MB 910 µA / 768KB |
| STM32H730 | 550 | 128 Kbytes | 564KB (incl.128K DTCM, 432KB Syst + 4K bckup) | • | • | TFT-LCD | SMPS[4] + LDO | 200 µA / 564KB 520 µA / 564KB |

La empresa ST Microelectronics, aparte de los microcontroladores en sí, vende también placas de evaluación con todos los periféricos necesarios y ofrece distintas herramientas de *software* para manejarlas o adaptar estas placas a nuestras necesidades. Entre la mucha documentación disponible están los proyectos de las placas en formato Altium [25], una de las herramientas de diseño electrónico asistido por ordenador (ECAD, *Electronics Coputer Aid Design*) más importantes en la actualidad. Altium dispone de versiones gratuitas para estudiantes, por lo que estos ficheros de diseño de las placas son una opción muy interesante por si en el futuro se quieren emplear para modificarla y adaptar el diseño a nuestras necesidades o simplemente para tener las medidas y posiciones exactas de cada componente y cada conector. El IDE del microcontrolador se llama STM32CubeIDE y es un editor muy avanzado que permite incluso depurar el *firmware* sin necesidad de herramientas adicionales. Su uso es algo más complicado que el de Arduino IDE pero existen muchos tutoriales y foros por internet donde consultar las dudas. A cambio de esta complejidad permite un control mucho mayor sobre la configuración del *hardware*, permitiendo definir un mapa de memoria propio, asignar variables o código a distintas áreas, tales como memoria flash, no volátil pero relativamente lenta, o RAM, volátil pero de mayor velocidad. Aparte de estas herramientas de tan bajo nivel, proporciona ejemplos para muchas placas de evaluación, con la configuración recomendada de todos estos elementos. Por tanto, si existiera alguna placa de evaluación de los modelos STMH747/757 o STMH745/755, que hemos visto que son muy similares, sería una opción interesante que estudiar. Volviendo a la página de ST vemos que las placas existen para los modelos STMH745/755 y se llaman NUCLEO-H745ZI-Q y NUCLEO-H755ZI-Q, siendo la única diferencia entre ellas las capacidades criptográficas, que dijimos que no necesitamos para este proyecto. Desafortunadamente no había disponibilidad en el momento de iniciar el proyecto, pero sí la hubo después y el precio de la placa era realmente competitivo, 26.39€. Al tener que realizar las simulaciones y avanzar en el diseño de la parte analógica, se pospuso la compra y no supuso un retraso en el proyecto.

Las primeras validaciones una vez recibida la placa indicaron unos tiempos de acceso a las señales digitales iguales a los del Arduino Portenta, aunque en este caso la configuración del reloj estaba mucho más a la vista y se pudo cambiar más fácilmente para llegar a los 480MHz en un núcleo y 240MHz en el otro. Los tiempos de manejo de los GPIO desde el núcleo M4 se redujeron a 4.2ns, manteniéndose en un único ciclo. La facilidad de cambio de la frecuencia de funcionamiento se debía a que existían ejemplos con esa configuración. Realmente la generación de esta señal no es muy intuitiva, tiene muchos parámetros a elegir y es necesario un buen conocimiento de la arquitectura del micro para conseguir el resultado deseado, salvo que se parta de una configuración previa como fue nuestro caso. La Figura 29 muestra una captura de pantalla de la interfaz de configuración donde se puede apreciar la gran cantidad de parámetros disponibles.

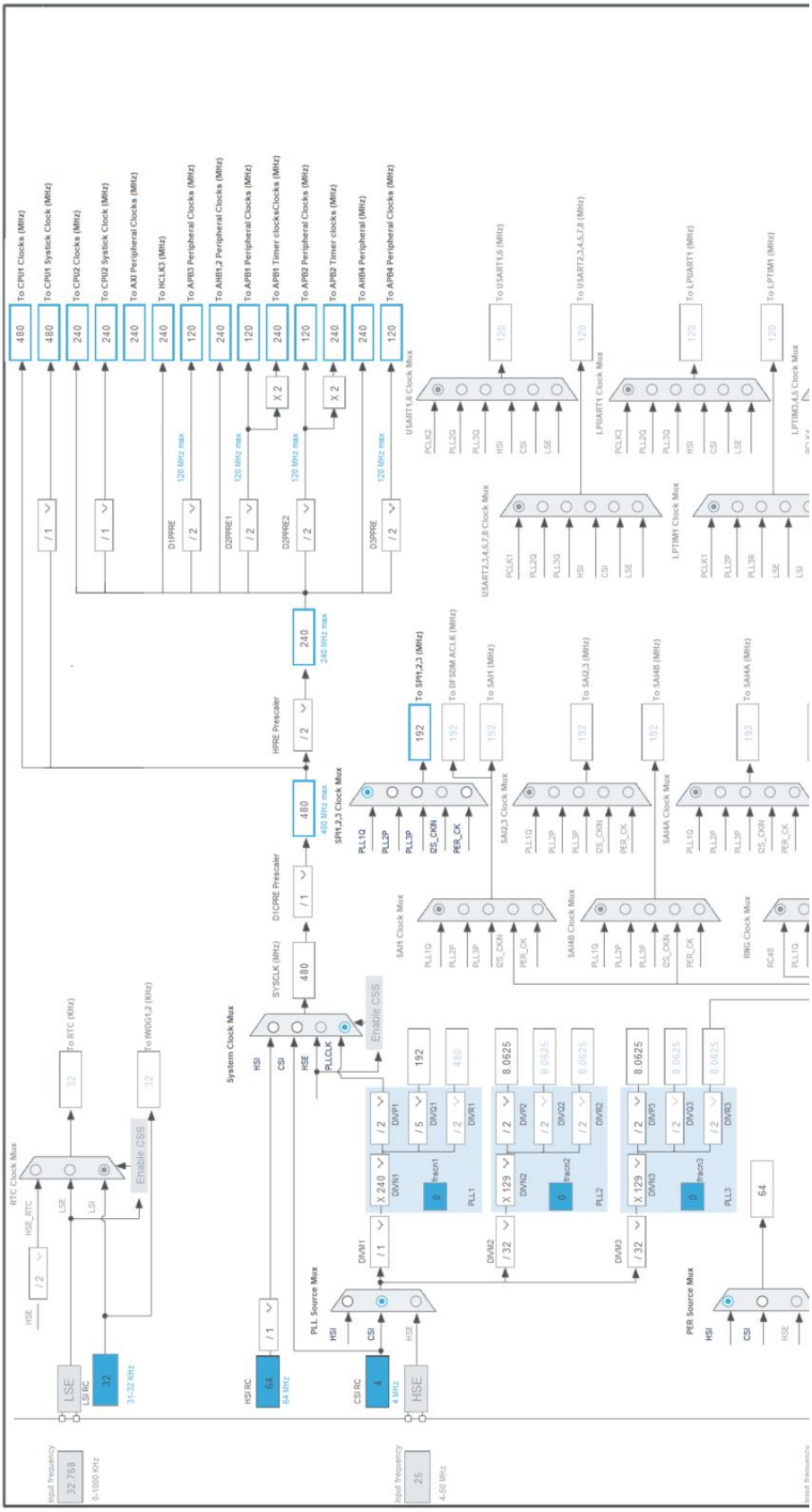


**Figura 29. Configuración del reloj en STM32CubeIDE**

Una vez que tenemos dos candidatos con potencia suficiente y tiempos de acceso pequeños, realmente muy parecidos, la decisión de cuál emplear se debe basar en detalles que faciliten el desarrollo o que puedan ofrecer alguna otra ventaja, aunque sea pequeña. Ya hemos justificado las ventajas e inconvenientes de cada entorno de desarrollo, Arduino es más sencillo para cosas básicas, pero probablemente más complejo si queremos acceder al micro a bajo nivel. La placa del Portenta es de dimensiones mucho más reducidas que la del NUCLEO-H755ZI-Q, que fue el modelo que estaba disponible para comprar. Lo que en principio es una ventaja, ya que un tamaño menor significa más sitio para el resto de los componentes, menores tiempos de propagación de señales, etc., se convierte en un inconveniente al realizar el trazado de pistas de la placa, ya que la densidad de ellas es mayor. En un proyecto en el que el tamaño de la placa no es un problema, como tampoco lo es el consumo, conectores más grandes con más espacio entre pines pueden ser una ventaja. Además, los pines de la placa de STM son los más comunes y baratos, tiras de pines de paso 0.1 pulgada (2.54mm), lo cual también es de agradecer. El último punto que puede ayudar a mejorar levemente la eficiencia es la distribución de las señales libres. El microcontrolador tiene muchas entradas y salidas de usuario, pero en ambas placas sus diseñadores debieron emplear algunas de ellas para determinadas funciones específicas y por tanto no están todas disponibles, y en general las que están disponibles pueden estar dispersas en distintos puertos. Por puerto en este caso nos referimos a cada conjunto de GPIO a los que se puede acceder de manera simultánea. El microcontrolador tiene hasta 11 puertos, dependiendo del encapsulado, que se identifican con las letras desde la A (puerto PA) hasta la K (puerto PK) y en general tienen un ancho de 16 bits, con alguna excepción. Idealmente nos gustaría conectar cada bus de datos del DAC o el ADC a uno de los puertos y quizá las señales de control a otro. Aunque luego discutamos sobre el ancho de palabra de cada conversor, es de esperar que al menos sean de 10 o 12 bits, por lo que sería interesante que al menos dos puertos tuvieran al menos 10 señales libres. A partir de la asignación de pines de los conectores del Arduino Portenta construimos la Tabla 5-4, en la que el número delante del guion bajo indica el conector y el de después el pin asignado. Se han excluido los pines usados por el bus USB, que emplearemos para la comunicación con el ordenador.

**Tabla 5-4. Distribución de los pines del Arduino Portenta por puertos**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PA | 1_34 | | | | 2_78 | | 2_80 | | 2_59 | 1_33 | 1_35 | | | 1_75 | 1_77 | |
| PB | | | | 1_63 | 1_65 | | 1_45 | 1_43 | 1_51 | 1_58 | | | | | 1_59 | 1_61 |
| PC | | | 2_74 | 2_76 | | | 2_61 | 2_63 | | | | | | 2_46 | | 2_48 |
| PD | | | | 1_56 | 2_50 | 2_52 | 1_55 | 1_57 | | | | | | | | |
| PE | | | | 2_54 | | | | | | | | | | | | |
| PF | | | | | | | | | | | | | | | | |
| PG | | | | 2_56 | | | | 2_65 | | 2_28 | 2_58 | 2_47 | 2_45 | | 2_26 | |
| PH | | | | | | | 2_68 | 1_46 | 1_44 | 2_16 | 2_14 | 2_12 | 2_10 | 1_49 | 2_8 | 2_62 |
| PI | 2_36 | 2_38 | 1_60 | 1_62 | 2_6 | 2_18 | 2_4 | 2_2 | 2_3 | 1_36 | 1_38 | 2_67 | | 1_40 | 1_37 | 1_39 |
| PJ | | | | | | | | 2_64 | 2_25 | 2_27 | 2_66 | | | | | |

En general podemos decir que los pines están bastante dispersos, salvo el puerto PI, por lo que para uno de los convertidores necesitaremos dos puertos y otros tantos accesos. El encapsulado del micro que monta la placa NUCLEO-H755ZI-Q tiene menos conexiones y consecuentemente menos entradas y salidas de usuario, en concreto las que se muestran en la Figura 30, obtenida del proyecto ECAD de la placa.

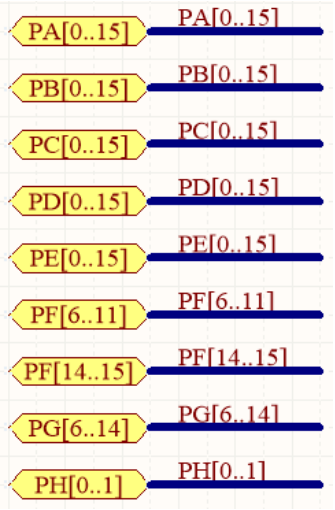



**Figura 30. GPIO de la placa NUCLEO-H755ZI-Q**

Todas estas señales están accesibles en conectores, aunque algunas ya tienen funciones asignadas y preferimos no usarlas salvo estricta necesidad para evitar conflictos. Los GPIO con funcionalidad asignada se recogen en la Tabla 5-5, donde vemos que los puertos PD y PE tienen muchas señales disponibles.

**Tabla 5-5. Pines con funciones asignadas en NUCLEO-H755ZI-Q**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PA | | ETH | ETH | | | | | ETH | USB | USB | | USB | USB | | |
| PB | LED1 | PB | | | | | | | | | | | | ETH | LED3 |
| PC | | ETH | | | ETH | ETH | | | | | | | | | OSC |
| PD | | | | | | | | | USB | USB | USB | | | | |
| PE | | LED | | | | | | | | | | | | | |
| PF | | | | | | | | | | | | | | | |
| PG | | | | | | | | USB | | | | ETH | | ETH | |
| PH | OSC | OSC | | | | | | | | | | | | | |

A la vista del análisis anterior decidimos usar la placa NUCLEO-H755ZI-Q, principalmente para poder sacar todo el partido a las herramientas de desarrollo del fabricante, pero también para simplificar el trazado de pistas de la placa que diseñemos con los demás elementos.

El microcontrolador incluye convertidores analógicos de alta frecuencia, que podemos plantearnos usar. En general cuando se diseñan circuitos analógico-digitales interesa separar físicamente lo máximo posible estos dos dominios, para evitar ruido

digital en las señales analógicas. Esto es realmente complicado si los convertidores analógicos están incorporados en el microcontrolador, por lo que los resultados al emplear sus DAC o ADC suelen ser peores que si se usan elementos externos. Aun así quisimos hacer la prueba, midiendo con un osciloscopio la salida del DAC con un valor constante. Vemos el resultado en la Figura 31, donde se aprecia un ruido considerable, de unos 10mVpp, inadmisible si vamos a emplear señales de poca amplitud.

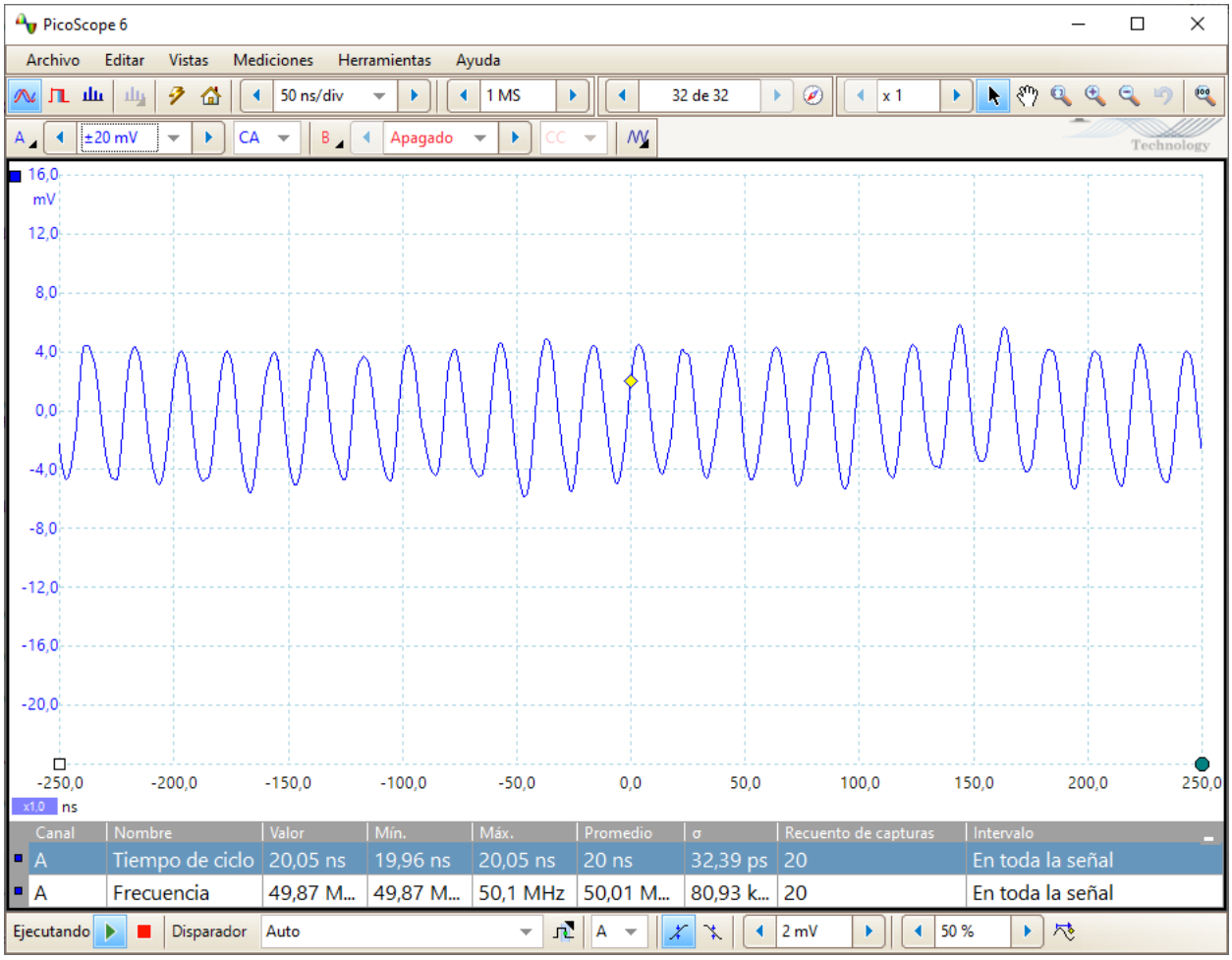



**Figura 31. Ruido del DAC interno del microcontrolador**

### 5.2.2 ADC

La elección del convertidor de analógico a digital ha sido mucho más sencilla que la del microcontrolador, basada en la interfaz paralela que hemos visto que ofrece mejores tiempos de comunicación, una resolución mínima de 12 bits y frecuencia de muestreo por encima de 1MSPS, que es el objetivo buscado, todo ello dentro del catálogo del fabricante y montador de las placas de circuito impreso. En el momento de comenzar el diseño solo había existencias de un componente que cumpliera con estas características, la referencia AD9240 de Analog Devices, un convertidor de 14 bits que ofrece una tasa máxima de 10MSPS, con interfaz paralela y un precio aproximado de 10€. El rango de entrada es de 0 a 5V, por lo que será necesario adaptar las ganancias

y niveles de tensión en la etapa analógica de entrada. El fabricante indica que los niveles lógicos son TTL (*Transistor-Transistor Logic*, lógica transistor-transistor), mientras que los del microcontrolador son LVTTL (*Low Voltage Transistor-Transistor Logic*, lógica transistor-transistor de bajo voltaje). Para las señales de datos esto no es un problema, se pueden emplear divisores de tensión resistivos o incluso confiar en las protecciones de la placa de evaluación, pero entre las señales de control del ADC está la de reloj, que según la hoja de características debe valer al menos 3.5V. Nuestra experiencia nos dice que, aunque estas sean las especificaciones que indica el fabricante muchas veces funciona con niveles de 3.3V, por lo que añadiremos al diseño un convertidor de niveles de tensión adecuado, pero también daremos la opción de llevar la señal directamente desde el micro, eligiendo la entrada mediante un puente físico. El convertidor de nivel elegido es el ampliamente utilizado TXB0104. La idea del selector es probar si realmente es necesario el adaptador de niveles y si no lo fuera eliminarlo de diseños futuros, si los hubiera.

### 5.2.3 DAC generador de la señal de excitación

Para la señal de excitación necesitamos un DAC que genere una señal sinusoidal con buena resolución. Si la frecuencia es la misma que la del ADC el diseño es más sencillo que si fuera distinta, por lo que buscaremos uno de características similares. Aparte de la frecuencia de la señal sinusoidal debemos poder variar también la amplitud, como mostramos en las simulaciones. Para conseguirlo hay varias alternativas, que podrían ser:

1. Emplear un DAC de mucha resolución, pongamos 20 bits y fondo de escala fijo, y generar la salida de manera directa, es decir, sin etapas analógicas posteriores. Tiene la ventaja de que la referencia puede ser muy precisa, incluso algunos modelos ofrecen referencias internas muy estables, pero a cambio, si la amplitud de la señal es muy pequeña, algo habitual, la resolución del DAC puede ser un problema, haciendo que la salida sea muy escalonada También encarece el diseño al ser un DAC de gran precisión y alta frecuencia. Descartamos esta opción.
2. Emplear un DAC con menos resolución, por ejemplo 10 bits o incluso 8, también con fondo de escala fijo y generar una señal siempre ajustada a este, para que la distorsión debida a la cuantificación sea pequeña. Posteriormente ajustar la amplitud con una etapa analógica de ganancia variable, por ejemplo, un DAC multiplicador que emplee esta señal como referencia.
3. Emplear un DAC similar al del punto anterior, pero con una referencia externa variable. De esta manera podemos controlar la amplitud de la señal generada de una manera sencilla variando el valor de la referencia, por ejemplo, con otro DAC.

La complejidad de las dos últimas soluciones es en principio similar, cualquiera de ellas nos valdría. La diferencia principal es que en la última solo necesitamos un DAC con el ancho de banda necesario para trabajar con señales de MHz y en la anterior dos. Por este motivo, y porque emplearemos otro DAC de más baja frecuencia para generar las señales de resultados y podemos reutilizarlo para esta función, preferimos la tercera

opción. Mirando los componentes disponibles elegimos la referencia AD5447 de Analog Devices, con un precio de aproximadamente 30€. Es un convertidor de 12 bits, también de bus paralelo, que funciona con alimentaciones entre 2.5 y 5.5V, admite una referencia de tensión externa de hasta ±10V y tiene una tasa máxima de actualización de 21.3MSPS.

### 5.2.4 DAC de resultados y referencia

Este componente es uno de los que hemos tenido que emplear por no haber disponibilidad del que realmente queríamos. Para mostrar los resultados al usuario de una manera rápida y precisa vamos a emplear señales analógicas, que permitirán una conexión sencilla a otros instrumentos de medida, como un osciloscopio, sistemas de adquisición de datos, etc. Emplearemos un DAC, o varios, para convertir los resultados, digitales, a analógicos. Idealmente este DAC debe tener poco ruido para no estropear los resultados de los cálculos, filtrados, etc. La tasa de actualización no es muy importante, bastaría con algún kHz o decenas de ellos, ya que la frecuencia de obtención de los datos vendrá limitada por los filtros aplicados, que estimamos tardarían al menos unos 10 periodos de la señal analizada, en el mejor de los casos. De proyectos anteriores conocíamos el DAC8734 de Texas Instruments, un DAC con interfaz SPI de 4 canales con muy bajo ruido y una referencia interna de 5V muy estable que permite configurarlo para generar salidas de ±5V o ±10V, que sería ideal para este proyecto. Lamentablemente la situación actual de desabastecimiento de componentes hacía que los plazos de entrega fueran superiores a 1 año en todos los distribuidores consultados, por lo que nos vimos obligados a buscar un reemplazo. El fabricante indicaba como reemplazo la referencia DAC81404, también de 4 canales, bus SPI, 16 bits de resolución, referencia interna y capaz de generar señales de ±10V, entre otras configuraciones posibles. La tasa de actualización depende de la frecuencia del bus SPI y del número de canales activos, pero en el peor de los casos es de varios cientos de kHz, adecuado para nuestras necesidades. Su precio es de unos 36€ aunque en el momento de escribir este documento también tiene problemas de disponibilidad. De los 4 canales emplearemos 3 para mostrar resultados al usuario, por ejemplo, las señales X, Y y R, y el cuarto como referencia para controlar la amplitud de la señal de excitación, como justificamos en el punto anterior.

### 5.2.5 Amplificador operacional

Tanto para la etapa analógica de entrada como para la de salida necesitaremos amplificadores operacionales. En la de entrada para aplicar alguna ganancia a la señal, filtrarla para eliminar la componente DC, adaptar las tensiones a las del ADC, etc. y en la de salida el propio DAC necesita un amplificador operacional como componente externo para la conversión. Por tamaño y precio intentaremos usar operacionales de dos canales, el diseño final incluye 4 amplificadores dobles. Entre las características que debemos buscar están el ruido, desde luego, pero también el ancho de banda y el *slew rate*, sobre todo si vamos a trabajar con señales de ±10V. La opción elegida es la referencia OPA2197 de Texas Instruments, que admite alimentaciones de hasta ±18V, anuncia 10MHz de producto ganancia·ancho de banda, 5.5nV/√Hz de densidad de ruido

por encima de 1kHz y 20V/µs de *slew rate*. Este valor se queda un poco corto para trabajar con señales de 500kHz y 10V de amplitud en las que la pendiente máxima es aproximadamente 31.4V/µs ($2 \cdot \pi \cdot A$), pero si realmente llegásemos a este límite significaría que los resultados han sido los mejores de entre los previstos y podríamos usar amplitudes de excitación más pequeñas si realmente quisiéramos probar el sistema a estas frecuencias. Su precio aproximado es de 1.50€/unidad. Las demás opciones que ofrecía el fabricante de las placas anunciaban peores características dinámicas, de ruido o precios mucho mayores.

### 5.2.6 Fuente de alimentación

Los distintos componentes analógicos necesitan alimentación de al menos ±12V y alguno 5V, mientras que los digitales demandan 3.3V y 5V. La placa del microcontrolador de puede alimentar mediante el cable USB que usaremos para configurarla y comunicarnos con el ordenador, y genera una señal de 3.3V que se puede emplear para alimentar elementos digitales. Los 5V provenientes de su entrada de alimentación también están disponibles en sus pines. Para las etapas analógicas emplearemos una fuente conmutada, más barata que las lineales y que será suficiente para validar el prototipo. Al ser externa, si viéramos que es un problema y que interesa emplear una lineal u otra conmutada de mejores características, se podrá reemplazar fácilmente. Hemos elegido la referencia GP25A13D-R1B de MEAN WELL, que proporciona 1A en 12V, 0.3A en -12V y 2.5A en 5V, valores más que suficientes para nuestros requisitos. Su precio ronda los 40€.

### 5.2.7 Otros componentes

Aparte de los componentes anteriores, que podemos decir que son los "principales", se necesitan muchos otros, como conectores, cables para alimentación o de datos, *jumpers* para las distintas opciones que dejamos configurables por ser un prototipo con componentes sin probar, etc. Los conectores de señal empleados normalmente en amplificadores lock-in modernos de alta frecuencia, son SMA (*SubMiniature version A*, Subminiatura versión A). En los antiguos o que no funcionan a tan altas frecuencias se suelen emplear BNC (*Bayonet Neil-Concelman*, Bayoneta Neil-Concelman), aún más comunes en muchos laboratorios y, en cualquier caso, más baratos, que son los que hemos elegido. La caja dependerá de las dimensiones finales del diseño. El conector de alimentación es de tipo DIN (*Deutsche Industrial Norms*, Normas industriales alemanas). Añadiremos unos relés para controlar la ganancia de la etapa de entrada, referencias de tensión donde sean necesarias, transistores para controlar los relés, diodos de protección, condensadores e inductancias para filtrar alimentaciones o señales, etc. No vamos a justificar la elección de cada uno de los componentes porque nos podríamos extender demasiado.

## 5.3 Diseño de la electrónica

El diseño electrónico se ha realizado empleando la herramienta Altium, que ya dijimos que tiene una versión gratuita para estudiantes y que el fabricante del módulo del

microcontrolador proporcionaba sus ficheros de diseño para este programa. Hemos aprovechado ese proyecto como punto de partida, eliminando todos los elementos innecesarios y manteniendo solo los conectores, de manera que el módulo encaje perfectamente sobre la placa desarrollada. A continuación comentamos algunos detalles de implementación de las distintas partes, sin reproducir los esquemáticos completos, que se podrán consultar en los anexos y la entrega digital.

### 5.3.1 Captura de esquemáticos

#### 5.3.1.1 Acondicionamiento de la entrada

El objetivo de un amplificador lock-in es analizar señales de corriente alterna, aunque hoy en día muchos son capaces de medir también valores de continua. Para eliminar esta componente no deseada filtramos la señal a analizar mediante un filtro pasa-alta. Hemos elegido la tipología Sallen-Key y calculado los componentes pasivos para una frecuencia de corte de 10Hz, que nos permitirá analizar sin distorsión señales a partir de aproximadamente 100Hz. Antes de este filtro realizamos la adaptación de impedancias de entrada mediante un seguidor de tensión, de manera que la entrada tenga alta impedancia. En sistemas de alta frecuencia es habitual adaptar tanto la impedancia de entrada como la de salida a 50Ω para evitar distorsiones en la señal y otros efectos no deseables. No consideramos que vayamos a llegar a frecuencias de trabajo tan altas, por lo que usaremos impedancias de entrada altas y de salida bajas. La Figura 32 muestra el esquema del filtro y la adaptación de impedancias.

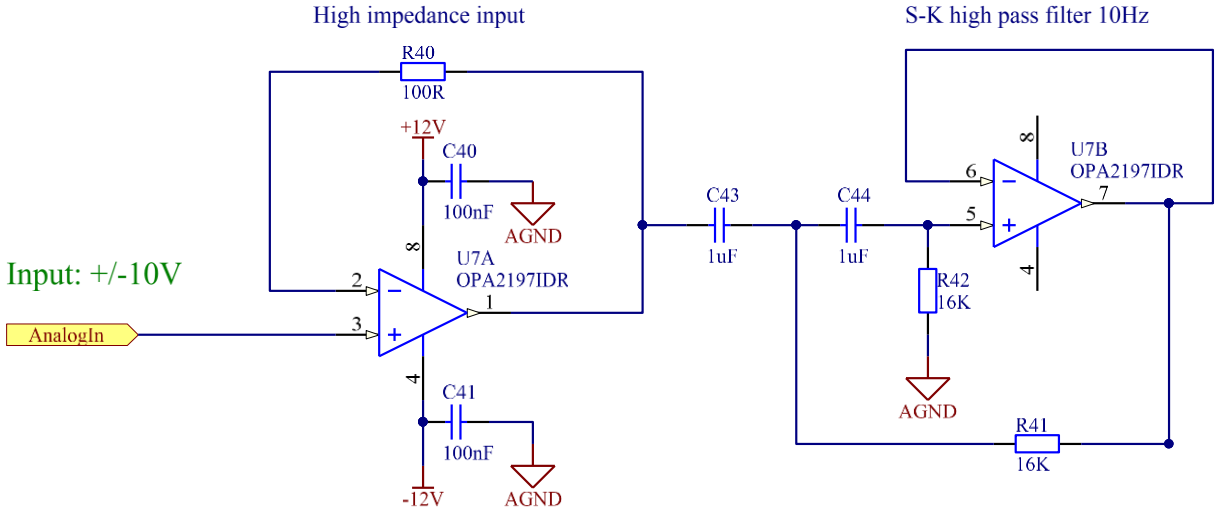



**Figura 32. Adaptación de impedancias y filtro de entrada**

La siguiente etapa en el camino de entrada es un amplificador de tensión de ganancia variable, que hemos implementado con otro amplificador operacional en configuración de amplificador inversor. Recordemos que la ganancia de esta configuración es igual al cociente entre la resistencia de realimentación y la de entrada, invirtiendo la fase. Empleamos en relé en cada una de estas ramas para poder cambiar los valores de ambas resistencias y obtener hasta 4 valores de amplificación según la

Tabla 5-6, suponiendo resistencias ideales. La tolerancia de las resistencias elegidas ha sido del 1%, por lo que no buscaremos precisiones superiores en los cálculos teóricos.

**Tabla 5-6. Ganancias de entrada**

| RL1 | RL2 | $R_{IN}$ (Ω) | $R_{FB}$ (Ω) | Ganancia |
|---|---|---|---|---|
| OFF | OFF | 10k | 9.91k | 0.991 |
| ON | OFF | 991 | 9.91k | 10 |
| OFF | ON | 10k | 1M | 100 |
| ON | ON | 991 | 1M | 1009 |

El diseño de este amplificador de muestra en la Figura 33.

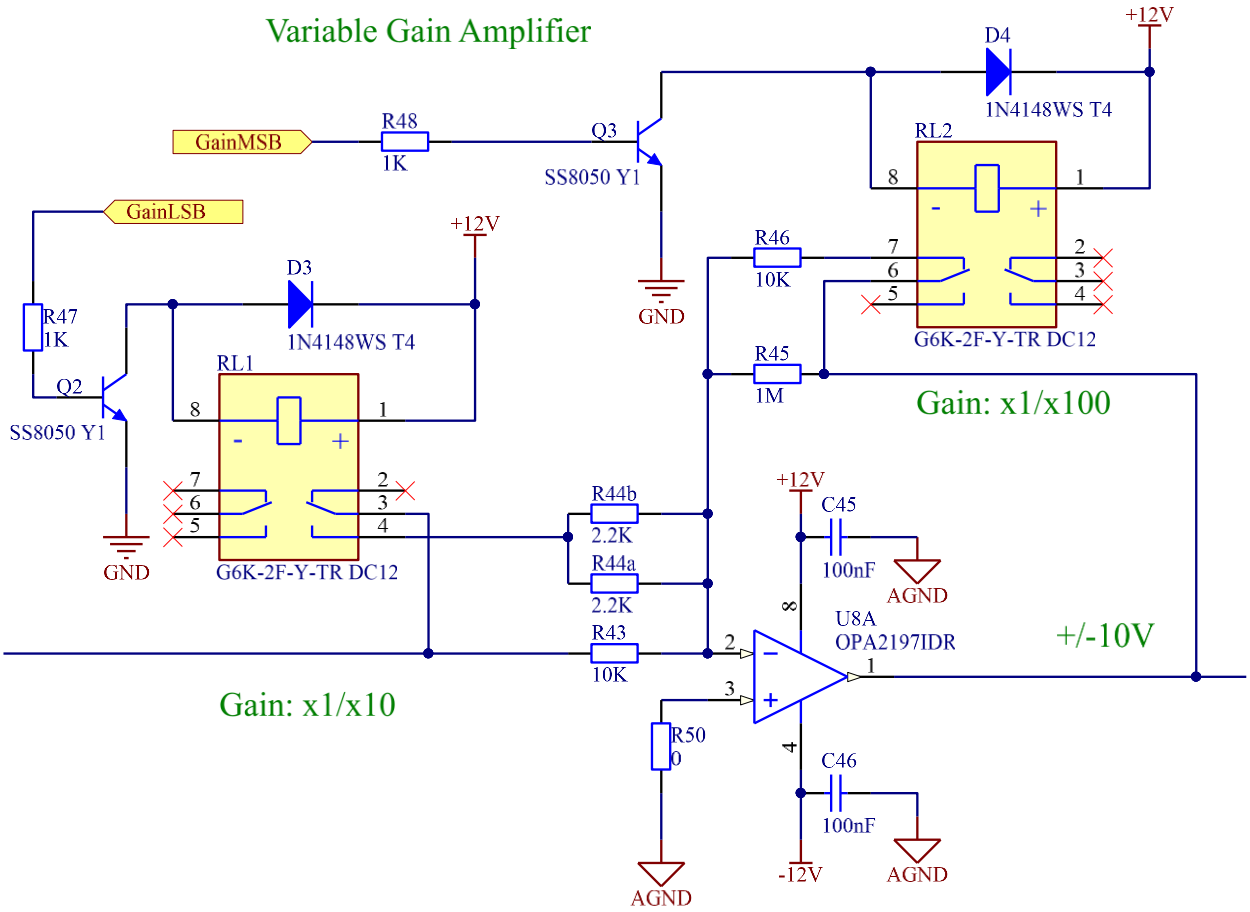



**Figura 33. Diseño del amplificador de ganancia variable**

A continuación se adaptan los niveles de tensión de ±10V al rango admitido por el ADC empleando el otro canal del amplificador operacional en configurador de restador con ganancia 0.25 sobre la señal amplificada y sumando 2.5V generados con una referencia de tensión. El diseño se puede consultar en los anexos.

### 5.3.1.2 ADC

El fabricante del ADC proporciona una serie de consejos en su hoja de características relativos a la selección de los componentes auxiliares y al filtrado y circuitería en general según la configuración empleada, que hemos seguido para realizar el diseño. En nuestro caso, que ya tenemos los niveles de tensión adaptados de la etapa anterior, hemos empleado su Figura 39, que reproducimos aquí como Figura 34. Hemos elegido el mayor rango de entrada para maximizar la relación señal/ruido.

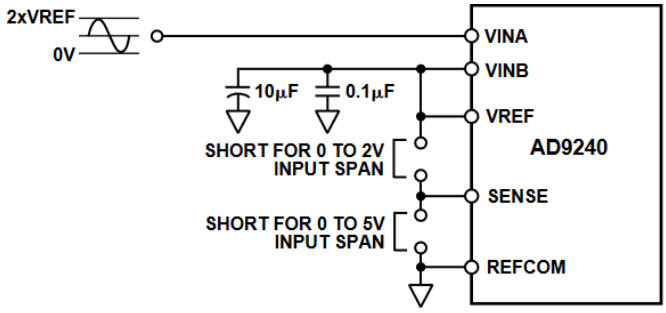



**Figura 34. Configuración del ADC para seleccionar el rango de entrada**

Podemos ver una versión simplificada del diseño en la Figura 35, en la que se incluyen los componentes pasivos recomendados por el fabricante.

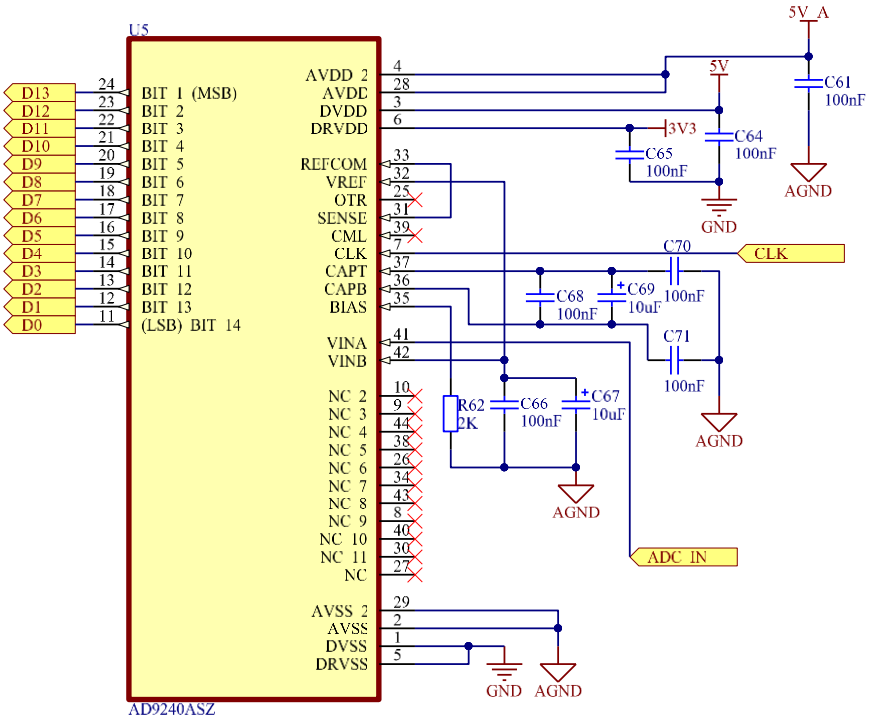



**Figura 35. ADC con componentes auxiliares**

### 5.3.1.3 DAC generador de la señal de excitación

El DAC elegido es del tipo multiplicador y de doble canal. En principio solo emplearemos uno de ellos y usaremos de referencia la señal generada por el DAC de 4 canales que también generará las salidas analógicas con los resultados de los cálculos, controlando así la amplitud de la señal de alta frecuencia. Al ser un primer prototipo y no haber trabajado nunca con ese modelo concreto que tiene muchos parámetros de configuración tanto *hardware* como *software*, hemos querido tener una segunda opción para controlar la amplitud de la señal, por si fallara la primera, que además nos suponía muy poco esfuerzo y coste adicional. Emplearemos el segundo canal de este DAC para poder generar la referencia de tensión del primero, de manera que mediante un puente podamos seleccionar cuál usar. Si finalmente fuera necesario usar esta opción habrá que poder elegir en el *firmware* qué canal actualizar mediante su línea de dirección y la actualización de la salida será un poco más lenta, al no ser fija esta señal, pero al menos tenemos una alternativa que nos permitiría validar el resto del diseño. Vemos la implementación en la Figura 36.

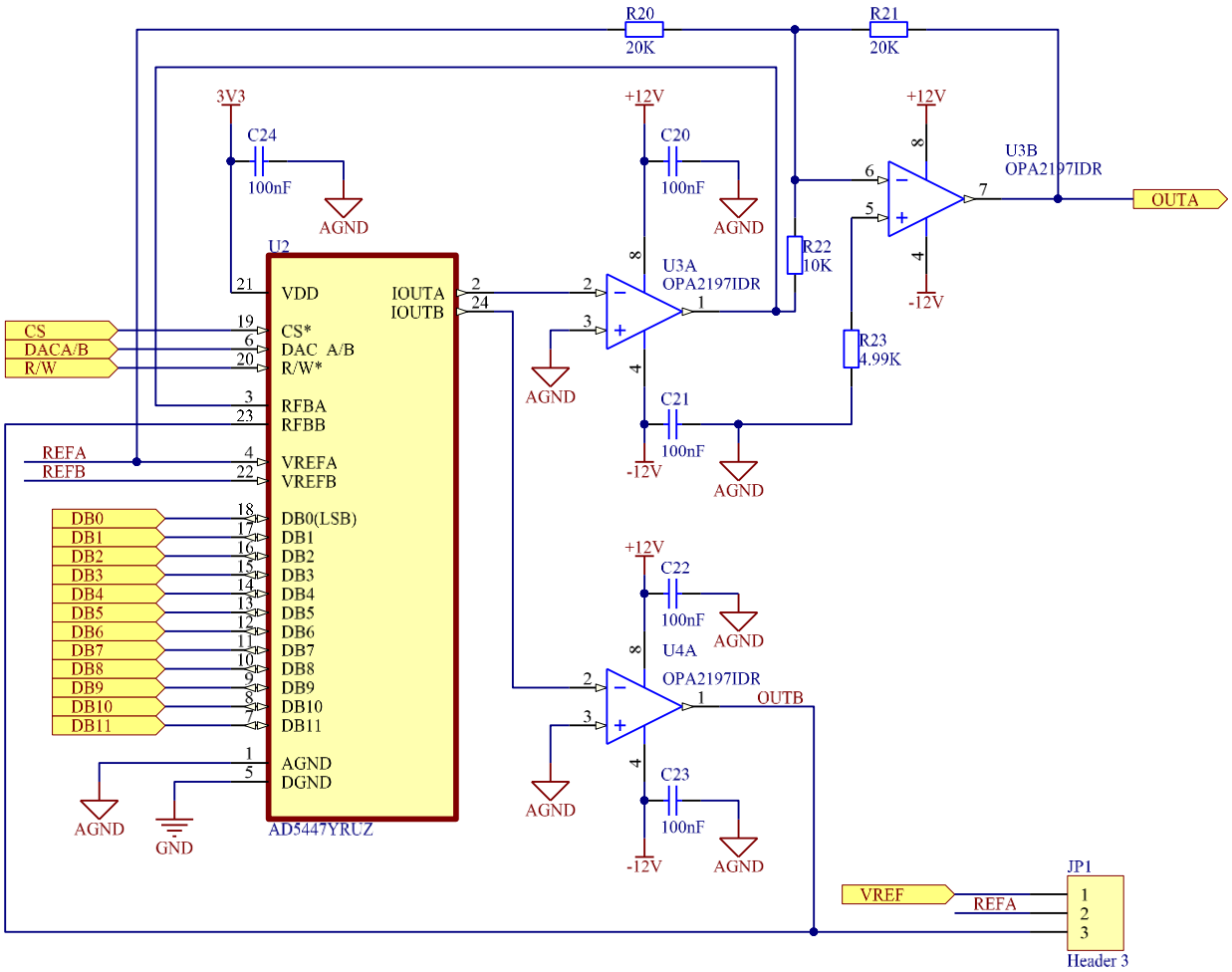



**Figura 36. Esquemático del DAC generador de la excitación**

### 5.3.1.4 DAC de resultados y referencia

La referencia DAC81404 permite múltiples opciones de configuración por software, mediante el acceso a registros, de las que aún no debemos preocuparnos. La configuración y actualización es mediante un bus SPI de hasta 50MHz, que al ser una señal de alta frecuencia hace recomendable el uso de resistencias para evitar distorsiones y rebotes de la señal. Normalmente estas resistencias se colocan en paralelo cerca del receptor o en serie cerca del emisor, que ha sido la opción que hemos elegido. El otro aspecto que hay que tener en cuenta, propio de este convertidor, es que incluye señales para medir la tensión en la carga, útiles para medidas a 4 cables, que en nuestro caso no tienen mucho sentido y uniremos a la salida siguiendo las sugerencias de la hoja de especificaciones. En cualquier caso, emplearemos pistas de cierta longitud por si no fuera esta la configuración adecuada, poder cortar las pistas en la placa. La Figura 37 muestra el diseño.

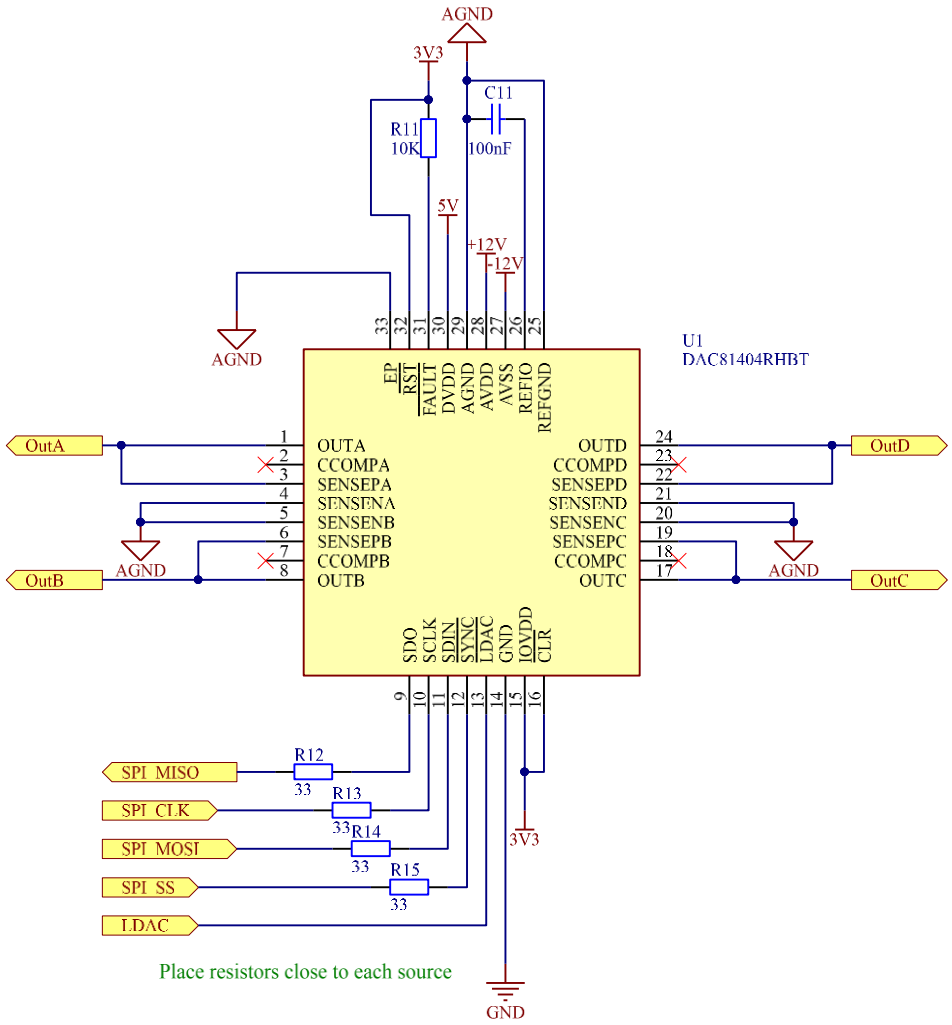



**Figura 37. Configuración *hardware* del DAC de resultados y referencia**

### 5.3.1.5 Otros componentes

Los epígrafes anteriores recogen los módulos principales, al menos desde el punto de vista funcional. Para que todo funcione correctamente hay que añadir conectores, filtrar las alimentaciones, etc. Algunos detalles de implementación del resto del diseño son:

1. Hemos añadido un LED por cada conexión necesaria para que el sistema funcione, es decir, uno para la alimentación analógica y otro para el cable USB. Estos indicadores irán en la parte frontal del aparato.
2. Hemos dejado varias opciones para seleccionar la señal de sincronización del ADC, ya que no se ha podido validar antes del diseño.
3. Hemos añadido al proyecto ECAD elementos tales como la fuente de alimentación, la caja o los cables, que no forman parte de la placa en sí, para que se incluyan automáticamente en la lista de material y facilitar la gestión de los documentos del proyecto.

La Tabla 5-7 muestra la asignación de los pines del microcontrolador a cada elemento, ordenada por puertos. En negrita aparecen las funciones propias del proyecto y sin resaltar las funciones asignadas por la placa de evaluación, que se han respetado para evitar conflictos.

**Tabla 5-7. Asignación de pines a cada elemento**

| | PA | PB | PC | PD | PE | PF | PG | PH |
|---|---|---|---|---|---|---|---|---|
| **0** | | LED1 | | **DAC0** | **ADC_Trigger** | | | OSC_IN |
| **1** | ETH | Push button | ETH | **DAC1** | LED2 | | | OSC_OUT |
| **2** | ETH | | | **DAC2** | **ADC0** | | | |
| **3** | | **CS_AD5447** | | **DAC3** | **ADC1** | | | |
| **4** | | **LDAC_81404** | ETH | **DAC4** | **ADC2** | | | |
| **5** | **SPI_CLK** | **SPI_MOSI** | ETH | **DAC5** | **ADC3** | | | |
| **6** | **SPI_MISO** | UART | | **DAC6** | **ADC4** | | | |
| **7** | ETH | UART | | **DAC7** | **ADC5** | | USB | |
| **8** | USB | **PWM** | | **USB** | **ADC6** | | | |
| **9** | USB | **GAIN_MSB** | | **USB** | **ADC7** | | | |
| **10** | | **GAIN_LSB** | | **USB** | **ADC8** | | | |
| **11** | USB | **R/W** | | **DAC8** | **ADC9** | | ETH | |
| **12** | USB | **DACA/B** | | **DAC9** | **ADC10** | | | |
| **13** | | ETH | | **DAC10** | **ADC11** | | ETH | |
| **14** | | LED3 | OSC_IN | **DAC11** | **ADC12** | | | |
| **15** | **SPI_SS** | | OSC_OUT | **SAMPLE** | **ADC13** | | | |

### 5.3.2 Emplazamiento de los componentes y trazado de pistas

Una vez terminada la captura de los esquemáticos el siguiente paso es el emplazamiento de los componentes y la conexión entre ellos, lo que se suele conocer como trazado de pistas o rutado. Este proceso, laborioso por naturaleza y muchas veces considerado un arte más que una ciencia, sí tiene ciertas reglas que es conveniente seguir y que muchas veces vienen especificadas en las propias hojas de características de algunos de los componentes. Por ejemplo, conviene filtrar las alimentaciones a la entrada, normalmente se usan ferritas y condensadores de gran capacidad, pero también en general cerca de los componentes electrónicos activos para poder satisfacer las demandas de corriente en momentos puntuales. Si la placa tiene cierta complejidad se recomienda emplear planos de masa o de alimentación para que estas líneas presenten impedancias muy pequeñas, distribuir las tierras en topología de estrella y en diseños que mezclan partes analógicas y digitales, separar todo lo posible estos dos dominios, tanto en señales como en alimentaciones y masas. Todas estas y otras recomendaciones, que sin duda hay que tener en cuenta al pasar el diseño a la placa, quizá no tengan mucho que ver con el contenido del máster y no vamos a entrar en ellas en mucho detalle. Simplemente queremos decir que hemos intentado seguirlas, incluso en etapas previas, como cuando elegimos convertidores externos en lugar de internos. El resultado ha sido una placa de 4 capas, las dos internas se han reservado una para masas, separando la analógica de la digital, y otra para alimentaciones. Las externas son las que llevan las señales, lo que nos permitiría a su vez hacer modificaciones sobre ellas más fácilmente si se detectase algún error de diseño. Para facilitar el montaje final se ha dimensionado la placa para medidas comunes, existiendo así mayor disponibilidad de cajas metálicas donde montar el prototipo. Al llevar dentro de la caja la placa del microcontrolador, de unos 133x70 $mm^2$ debemos irnos a unas medidas suficientes para que quepan este módulo, los conectores y el resto de los elementos. La mayoría de los fabricantes de cajas para placas electrónicas pasan de 100mm a 160mm en la longitud y ancho de las placas a albergar, por lo que hemos diseñado la placa de 160x160$mm^2$. En la Figura 38 mostramos un detalle de la zona de mayor densidad de componentes coloreada según la funcionalidad. Vemos que las etapas analógicas de entrada y salida ocupan una proporción importante, pese a ser muy simples, lo que justifica el uso de versiones digitales.

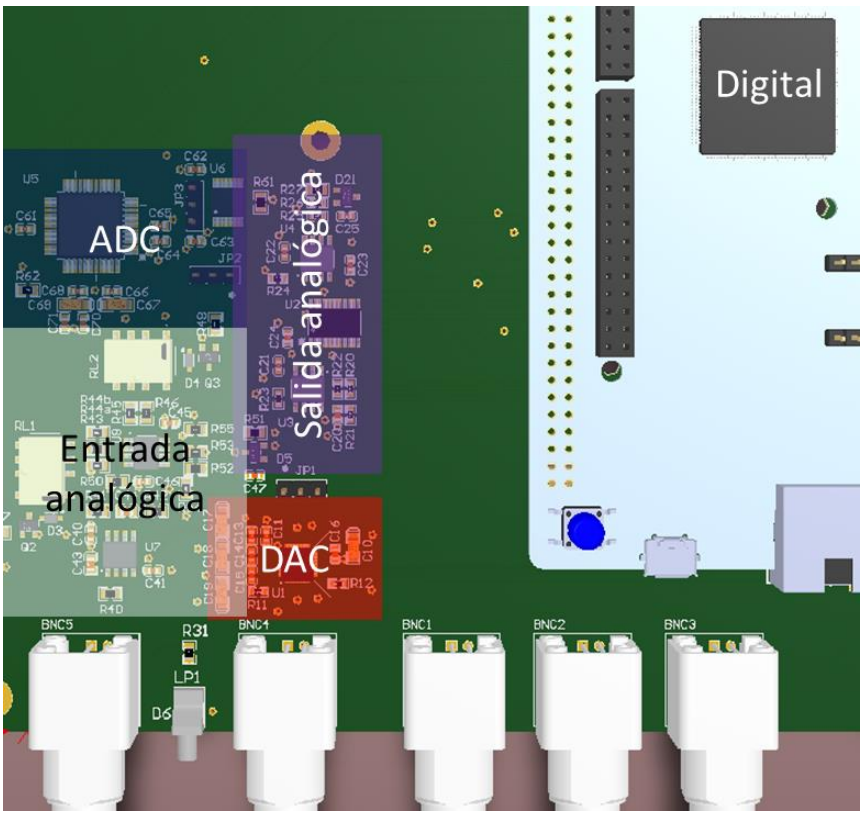



**Figura 38. Distribución de los componentes por funcionalidad**

### 5.3.3 Diseño mecánico

Hemos aprovechado la misma tecnología de fabricación de placas de circuito impreso para diseñar las carátulas (Figura 39 y Figura 40), de manera que no necesitemos herramientas de mecanizado convencional para montar el prototipo. En los últimos años, debido en parte al auge de la iluminación con LED y sus necesidades de disipación de calor, muchos fabricantes de PCB (*Printed Circuit Board*, placa de circuito impreso) incluyen la posibilidad de fabricar las placas en aluminio a buen precio, que aparte de buen conductor térmico lo es también eléctrico, lo que ayuda a aislar la electrónica de interferencias electromagnéticas. Emplearemos este material en la fabricación.

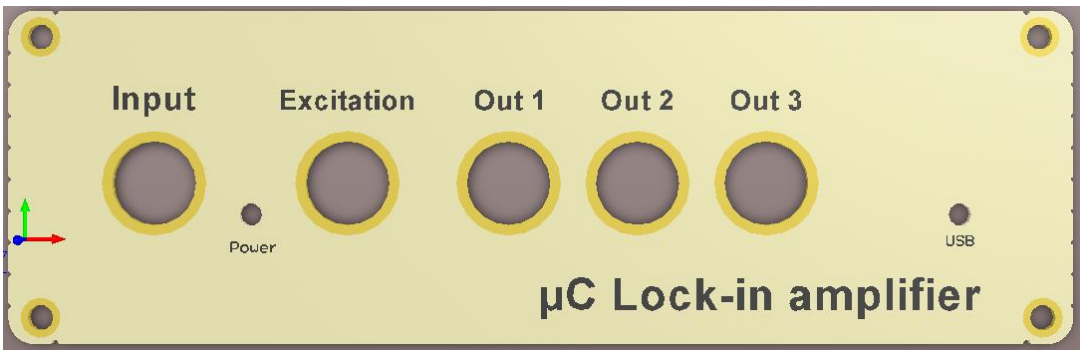



**Figura 39. Diseño de la carátula frontal**

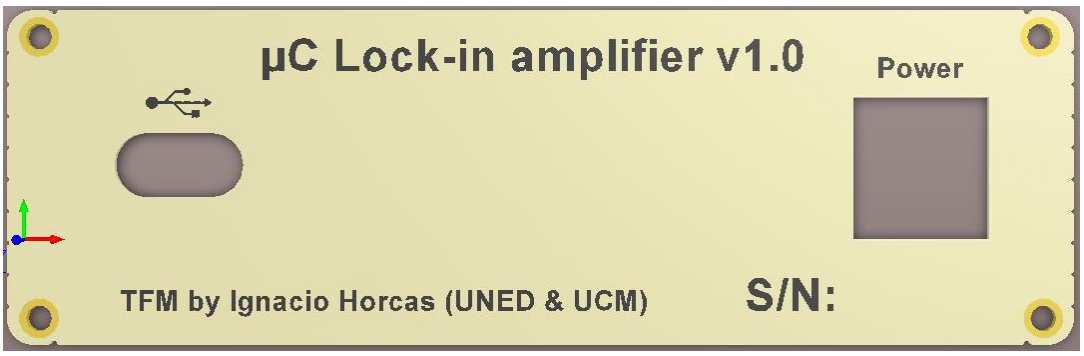



**Figura 40. Diseño de la carátula trasera**

La composición final queda como se muestra en la Figura 41, en la que la caja se muestra semitransparente para apreciar algunos detalles de su interior. Más adelante mostraremos fotografías reales del resultado final.

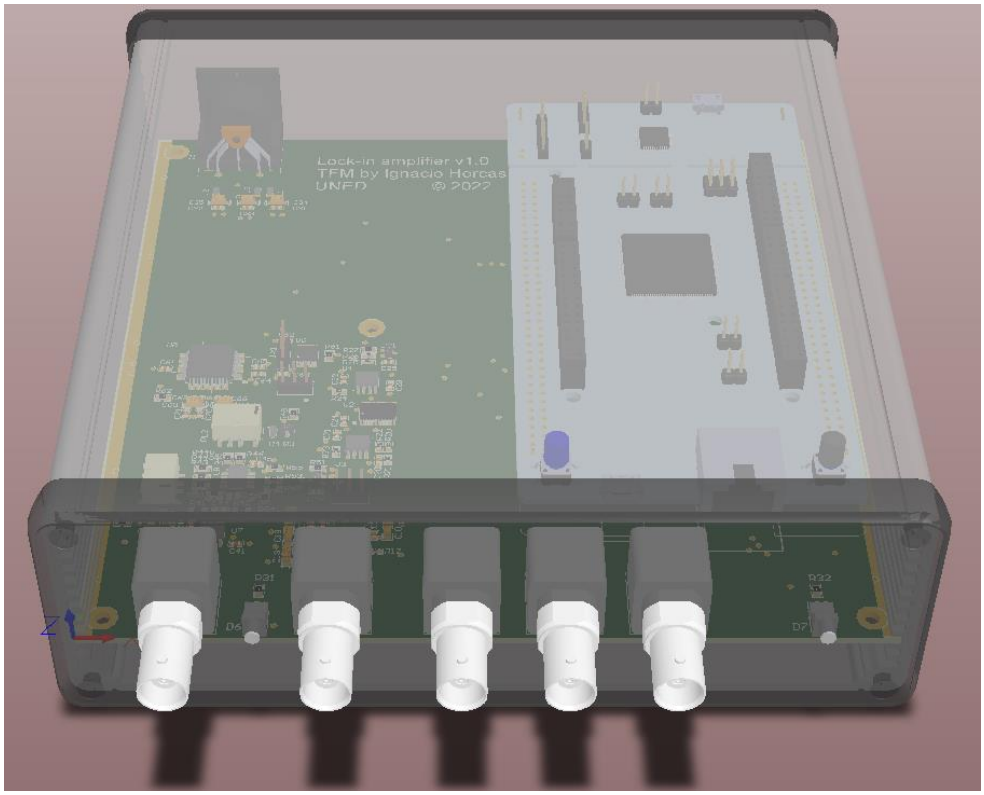



**Figura 41. Diseño *hardware* final**

## 5.4 Firmware

Entendemos por *firmware* el programa informático que lleva incorporado un sistema empotrado, en este caso el amplificador lock-in. En nuestro proyecto podemos clasificar las tareas que debe realizar en las siguientes categorías:

1. Control de los elementos hardware.
2. Cálculos y otras tareas específicas del amplificador lock-in.
3. Control de la interfaz de usuario y comunicaciones con otros dispositivos.

El microcontrolador empleado dispone de gran cantidad de *hardware* dedicado a tareas muy concretas, que ayudan a mejorar la eficiencia del código descargando la CPU. Algunas son habituales en este tipo de sistemas, como los módulos de comunicación con distintos tipos de periféricos (UART, SPI, $I^2C$, etc.), distintos canales de DMA o temporizadores, y otras son menos habituales, como un juego de instrucciones ampliado con ciertas operaciones muy concretas. Su uso, por supuesto, es opcional, las mismas tareas que realizan estos módulos por *hardware* se pueden realizar por *software*, de una manera mucho menos eficiente, eso sí. En la medida de lo posible se han intentado usar estos recursos para crear un código que se ejecute más rápido, pero no se ha llevado hasta el último nivel. Téngase en cuenta que en general el uso de cada una de estas herramientas necesita una configuración previa, costosa en tiempo de desarrollo e incluso en tiempo de ejecución, haciendo que en algunos casos la ganancia

de tiempo en la comunicación, por ejemplo, sea menor que lo que se tarda en configurar los registros adecuados para llevarla a cabo.

Al igual que ocurre con otros fabricantes, la familia de controladores de ST es muy amplia, con diferencias muy sutiles entre algunos de sus modelos. Una consecuencia de esto es que hay funcionalidades *hardware* muy específicas que pueden estar disponibles en un modelo y no en otro y a veces es complicado determinar si, por ejemplo, una determinada instrucción del *set* está implementada por *hardware* o no en nuestro modelo concreto. La documentación que proporciona el fabricante es muy extensa, como debe ser en dispositivos de esta complejidad, por lo que buscar la información exacta muchas veces es una labor tediosa. A modo de ejemplo, la hoja de características del microcontrolador tiene una extensión de 251 páginas y el manual de referencia (RM0399) son 3528 páginas, que se complementan con distintas notas de aplicación, notas técnicas, videotutoriales, códigos de ejemplo, foros de debate, etc. A veces resulta complicado encontrar la información buscada, no porque no exista, sino porque no sabemos si debemos buscar en una nota de aplicación, en una nota técnica, en la hoja de características o en el manual, o porque en un vídeo habla de algunas instrucciones específicas del juego de instrucciones de un determinado núcleo de un microcontrolador y no está claro si se aplica a otros de la misma familia, etc. Un ejemplo concreto de esto ha sido el uso de una instrucción para compactar datos en un ciclo de reloj (PKHBT), que escribe en un registro de 32 bits los 16 menos significativos de dos operandos que recibe. Esta instrucción, muy útil al comunicar datos de 16 bits, no está implementada por hardware en el micro empleado, pero hasta que no la utilizamos, medimos tiempos y consultamos el código de la biblioteca de acceso al *hardware* para ver que la implementación realmente era por *software* y tardaba lo mismo que si realizamos la operación de una manera más clara, no vimos que realmente no era tan útil. Algunos controladores que implementan núcleos Cortex-M7 incluyen por hardware módulos de filtrado digital de señales, que probablemente reducirían el tiempo de cálculo, pero analizando con detalle las características del nuestro, parece que el nuestro no los incluye. A la vista de estas experiencias, decidimos no intentar optimizar el código fuente hasta su nivel máximo empleando instrucciones propias de ensamblador hasta tener la funcionalidad completa y evaluar los resultados obtenidos.

Esta variedad en el catálogo tiene otra desventaja, y es que los ejemplos que proporciona el fabricante, tan útiles para comenzar un proyecto con una tecnología a la que no se está muy acostumbrado, están disponibles para algunos de los modelos, pero no adaptados a todos, aunque el micro fuera perfectamente capaz de ejecutarlos. Pero quizá haya que cambiar el mapa de memoria o alguna instrucción específica que haga que no se pueda importar y empezar a usar o simplemente puede darnos miedo arriesgarnos a dañar la placa si no es compatible. Como indicamos antes, el modelo de placa que hemos usado es el NUCLEO-H755ZI-Q, que directamente no aparece en el listado de ejemplos del IDE del fabricante, como se puede apreciar en la Figura 42. La que sí aparece es la H745, que es similar y los ejemplos son 100% compatibles, que son los que hemos usado.

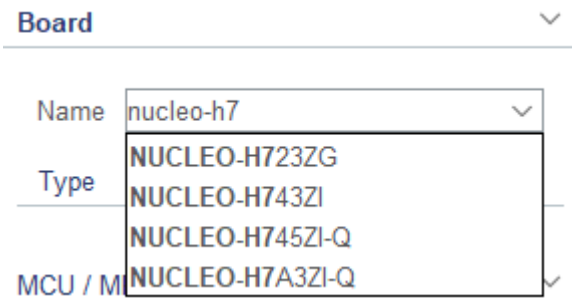



**Figura 42. Placas con ejemplos disponibles de la familia NUCLEO-H7**

El entorno de desarrollo permite escribir el código en C o en C++, incluyendo instrucciones de ensamblador como funciones intrínsecas. La opción de escribir todo el proyecto en ensamblador no llegamos ni siquiera a plantearla dada la complejidad y extensión estimada del código fuente. El inconveniente de programar en lenguajes de alto nivel es que perdemos el control sobre muchos aspectos y la eficiencia del código depende del compilador y de su configuración. En unas primeras pruebas observamos que en la configuración por defecto esa eficiencia era mucho menor que la esperada. Recordemos que el IDE dispone de una herramienta de depuración de código, con inspector de variables, puntos de ruptura, ejecución por bloques, etc. Muchas de las optimizaciones posibles complicarían las labores de depuración, por ejemplo, si se eliminan variables, se reestructura el código para cambiar el orden de ejecución, etc. Una de las opciones que permite elegir el IDE es el nivel de optimización, que hemos establecido al máximo, consiguiendo tiempos de ejecución mucho mejores incluso en ejemplos sencillos como activar o desactivar una señal digital.

### 5.4.1 Distribución de las tareas entre núcleos

Para sacar todo el potencial a un procesador de dos núcleos debemos emplear ambos, comunicarlos de manera eficiente y limitar el número de estas comunicaciones, que será tiempo en el que ninguno de los dos núcleos se podrá dedicar a otras tareas, como cálculos o actualización de periféricos. Si la cantidad de datos a transmitir fuera necesariamente muy grande se podría emplear DMA para aligerar la carga de la CPU, incluso configurándolo para transmisiones continuas de un *buffer* circular para no tener que reconfigurarlo en cada envío. En ese caso habría que tener más cuidado con la sincronización entre núcleos y estas comunicaciones para asegurarnos de que han llegado todos los datos antes de leerlos o de que ya están puestos en el *buffer* antes de transmitirlos.

Hemos visto que por la arquitectura de buses internos el acceso a los GPIO es más rápido desde del núcleo M4 y los cálculos en general lo serán en el M7, ya que tiene el doble de frecuencia de reloj y más recursos *hardware*, incluyendo una unidad de coma flotante de doble precisión. Por tanto, interesa que el primero se encargue de las tereas en tiempo real estricto, deshabilitando sus interrupciones y WDT y el segundo de los cálculos en general y atención a rutinas. Como veremos en el epígrafe 5.4.2, hemos diseñado el código fuente de manera modular, intentando encapsular muchas funciones que bien podrían ejecutarse en cualquiera de los dos núcleos. Esto nos permite realizar

pruebas de distintas distribuciones entre ellos de manera sencilla, simplemente cambiando de lugar la llamada a una función. Si estas funciones son *inline* además no habrá penalización de tiempos por cambios de contexto.

El núcleo M4 debe ejecutarse en tiempo real estricto, idealmente con un bucle que se repita de manera continua con un periodo inferior a 1μs. Queremos que el tiempo de ejecución del bucle sea lo más estable posible, porque en caso de no serlo deberemos configurar su periodo al tiempo máximo y perderemos eficiencia en las ejecuciones que tarden menos. Las tareas que no se ejecuten en cada ciclo de muestreo, como la actualización del DAC de resultados o la comunicación por USB, por tanto, son más adecuadas para el núcleo M7. En otras tareas la asignación al mejor núcleo no está tan clara y dependerá de circunstancias que no conozcamos hasta la implementación y sus pruebas. Por ejemplo, una vez adquiridos los datos del ADC, la primera operación que se debe hacer con ellos es multiplicarlos por la señal I (seno) y Q (coseno) y posteriormente aplicar un filtro IIR a los resultados (X e Y). El núcleo M7 es más eficiente realizando estas operaciones, pero si le asignamos ambas tareas podría suceder que la carga de trabajo esté tan desequilibrada que sea el que limite la frecuencia del ciclo, aunque tenga más potencia, o que la latencia o el ancho de banda de las comunicaciones entre núcleos no sea suficiente y haya que disminuir la frecuencia de trabajo global. En el primer caso se podrían redistribuir mejor las tareas, quizá dejando para el M7 solo el filtrado y realizando los productos por I y Q el M4 y transmitiendo las señales X e Y sin filtrar. En el segundo, habría que reducir las comunicaciones, que serían el cuello de botella, quizá implementando un filtro FIR en el propio núcleo M4 y enviando solo el promedio de un ciclo o de varios de ellos. Según estos criterios, identificamos las tareas de la Tabla 5-8 y su posible asignación a cada uno de los núcleos, marcada con una X cuando tenga sentido y dejando la casilla en blanco cuando no. Tras las pruebas realizadas al terminar la implementación, observamos que la mejor distribución es aquella en la que el núcleo M4 se encarga solo del manejo de los convertidores de tiempo real y el M7 realiza todas las demás tareas, incluyendo el producto del dato leído por las señales I y Q, obteniendo tiempos de ciclo de unos 140ns. Este tiempo representa 34 ciclos de reloj del núcleo M4 y 68 del M7, valores muy bajos, sobre todo el primero, dado que también incluye el manejo de los convertidores analógicos. La distribución final se indica en la misma tabla marcando en negrita la tarea asignada a cada núcleo.

**Tabla 5-8. Tareas asignables a cada núcleo**

| Tarea | Cortex®-M7 | Cortex®-M4 |
|---|---|---|
| Configuración de periféricos | **X** | X |
| Manejo del DAC y ADC en tiempo real | | **X** |
| Multiplicación de la muestra por I y Q | **X** | X |
| Filtro IIR | **X** | X |
| Filtro FIR | **X** | X |
| Actualización del DAC de resultados | **X** | |
| Comunicación con el PC | **X** | |
| Configuración de parámetros de funcionamiento (ganancia, frecuencia, fase de referencia, amplitud, …)[3] | **X** | |
| PLL[4] | X | |

### 5.4.2 Estructura del código

El código se ha desarrollado en lenguaje C, intentando seguir las buenas prácticas que facilitan su desarrollo y mantenimiento, como separar el código en ficheros según su función, y adaptando otras a las características de los sistemas embebidos, donde, por ejemplo, la cantidad de memoria disponible es más limitada que en otros sistemas y muchas veces prima más la eficiencia que la claridad y mantenibilidad. El DAC y ADC paralelos, por su simplicidad, se manejan directamente desde un bucle que se ejecuta continuamente en el núcleo M4 y su tarea principal es esta, actualizar la salida en tiempo real, generando una señal sinusoidal, y leer la respuesta del sistema. Los demás elementos tienen sus propios ficheros .c y .h, que se muestran en la Tabla 5-9. Aparte de estos ficheros que hemos implementado, el proyecto incluye otros que ya existían en los ejemplos de partida.

[3] Excepcionalmente, en caso de que no se consiga controlar la amplitud con DAC inicialmente pensado para ello y se tenga que hacer con el segundo canal del que genera la señal de excitación, esta tarea debe ir en el núcleo M4 para no manejar el mismo dispositivo desde ambos núcleos y evitar conflictos.

[4] En el primer prototipo no se ha implementado el bucle de control de fase (PLL), pero si se quisiera añadir en una versión posterior el *hardware* sería compatible y se podría modificar el *firmware* para incluirlo. En ese caso, debería ir en el núcleo indicado.

**Tabla 5-9. Ficheros de código fuente del firmware**

| Fichero | Descripción |
|---|---|
| main.* | Contiene el código específico de cada núcleo, con la función que se ejecutará al inicio del programa en cada uno de ellos y otras funciones específicas de cada unidad |
| DAC81404.* | Controla el DAC81404, incluyendo la configuración del bus SPI. Actualiza los valores empleando interrupciones, por lo que debe ir en el núcleo M7 |
| AD5447.h | Contiene definiciones de pines del DAC AD5447 |
| filter.* | Implementa los filtros digitales que se aplicarán a las señales |
| lockin.* | Incluye inicializaciones propias del amplificador lock-in, funciones que controlan sus parámetros de funcionamiento y el bucle que se ejecutará de manera continua en el núcleo M4 |
| pwm.* | Controla señales que se utilizan como reloj para otros periféricos |
| sharedMemory.* | Define una estructura para compartir datos entre núcleos y controla su gestión en memoria física accesible por ambos |
| uart.* | Controla el puerto serie y los comandos de control del amplificador |

### 5.4.3 Memoria

Los distintos ejemplos que proporciona el fabricante vienen con un mapa de memoria propio para cada núcleo, que son los que hemos usado y resumimos en la Tabla 5-10. En ella podemos ver que cada núcleo tiene sus zonas de memoria asignadas y que no solapan entre ellas, pero si vamos al manual de referencia del controlador encontramos otros bloques, accesibles por ambos (Tabla 5-11), que pueden ser usados como memoria compartida para intercambiar datos entre los dos núcleos, lo que evita la necesidad de usar RPC. El tiempo de acceso depende del bloque concreto y a qué bus esté conectado. Se han hecho pruebas buscando los tiempos más rápidos desde el núcleo M4, obteniendo los mejores resultados con el bloque SRAM2.

**Tabla 5-10. Mapa de memoria por defecto**

| Núcleo | Flash | | RAM | |
|---|---|---|---|---|
| | Inicio | Longitud | Inicio | Longitud |
| **M7** | 0x08000000 | 1024K | 0x20000000 | 64K |
| **M4** | 0x08100000 | 1024K | 0x10000000 | 288K |

**Tabla 5-11. Mapa de memoria RAM común**

| Region | Boundary address | Arm® Cortex®-M7 | Arm® Cortex®-M4 | Type | Attributes | Execute never |
|---|---|---|---|---|---|---|
| RAM | 0x38801000 - 0x3FFFFFFF | Reserved | | Normal | Write-Back Write Allocate | N |
| | 0x38800000 - 0x38800FFF | Backup SRAM | | | | |
| | 0x38010000 - 0x387FFFFF | Reserved | | | | |
| | 0x38000000 - 0x3800FFFF | SRAM4 | | | | |
| | 0x30048000 - 0x37FFFFFF | Reserved | | | | |
| | 0x30040000 - 0x30047FFF | SRAM3 | | | | |
| | 0x30020000 - 0x3003FFFF | SRAM2 | | | | |
| | 0x30000000 - 0x3001FFFF | SRAM1 | | | | |
| | 0x24080000 - 0x2FFFFFFF | Reserved | | | | |
| | 0x24000000 - 0x2407FFFF | AXI SRAM | | | | |
| | 0x20020000 - 0x23FFFFFF | Reserved | | | | |
| | 0x20000000 - 0x2001FFFF | DTCM | Reserved | | | |

Hemos aprovechado este bloque de memoria compartida para definir una estructura que servirá para intercambiar datos entre núcleos y sincronizarlos en caso de necesidad. Incluirá un buffer circular donde ir guardando las muestras adquiridas por el ADC junto con la salida del generador de fase (θ en el aparatado 3.2.1) de cada una de ellas, para poder calcular las señales I y Q necesarias para el proceso de demodulación. Cada uno de estos valores se guarda como un entero de 16 bits, ocupando una única palabra de 32 bits entre ambos y reduciendo el número de accesos a la memoria compartida. El tamaño del *buffer* debe ser suficiente para que cuando el núcleo M7 esté realizando otras tareas, como actualizar el DAC de resultados o manejar el puerto serie, no se desborde. Se ha añadido también código de control sobre el uso máximo de *buffer* para comprobar que este problema no sucede y se ha dimensionado de manera que en las pruebas nunca ha superado la mitad de su capacidad. Los campos que tendrá la estructura de la memoria compartida se recogen en la Tabla 5-12.

**Tabla 5-12. Estructura de datos de la memoria compartida**

| Nombre | Tipo | Descripción |
|---|---|---|
| sync | uint32_t | Variable para sincronizar los núcleos |
| phaseInc | uint32_t | Incremento de fase en cada ciclo |
| lockInPhase0 | uint32_t | Desfase de la señal de excitación |
| phaseBuffer | uint32_t[] | *Buffer* circular con las muestras medidas y su fase |
| lastSampleIndex | uint32_t | Índice de la última muestra almacenada en el *buffer* |

### 5.4.4 Filtrado digital

Aunque en las simulaciones se han empleado filtros IIR y FIR sobre las señales X e Y, en la implementación del primer prototipo solo se han incluido de respuesta infinita, que son los más habituales en este tipo de sistemas. La modificación para añadir filtros de respuesta finita a un impulso se puede realizar actualizando solo el *firmware*, si en el futuro se considerase adecuado. El filtro empleado es de tipo Butterworth de segundo orden y los cálculos se realizan en el núcleo M7, que dispone de una unidad en coma flotante de doble precisión implementada en *hardware*. Este dato podría ser irrelevante, cualquier operación en aritmética flotante se puede realizar también en aritmética fija, y la comunidad de desarrolladores de programas de procesado de señales, o de *firmware* en general, está dividida entre quienes sostienen que la aritmética en coma flotante es totalmente innecesaria y quienes la consideran útil al menos en algunos casos. Sin querer entrar en esa discusión, aquí la hemos empleado, y vamos a justificar por qué. Nuestra primera idea fue realizar todas las operaciones en coma fija, incluso en el núcleo M4, y definimos tipos de datos de 16 bits en los que el primero era el bit de signo y los otros 15 la parte decimal, sin parte entera. Podía parecer una buena opción si los datos del ADC venían en esa representación de una manera natural, normalizados al fondo de escala del convertidor y un filtro IIR no es más que una serie de productos y sumas, siendo el resultado final esperado algo menor que la entrada, ya que se eliminan las componentes de alta frecuencia. Por muy eficientes que sean los cálculos en coma flotante, lo primero que necesitamos es convertir los datos desde la representación en coma fija, que también consume algún ciclo. La nota de aplicación AN4044 de ST muestra los ciclos necesarios para realizar algunas operaciones en coma flotante.

**Tabla 5-13. Ciclos necesarios para ejecutar algunas operaciones en coma flotante en M4/M7**

| Instruction | Description | Cycles |
|---|---|---|
| VABS.F32 | Absolute value | 1 |
| VADD.F32 | Addition | 1 |
| VSUB.F32 | Subtraction | 1 |
| VMUL.F32 | Multiply | 1 |
| VDIV.F32 | Division | 14 |
| VCVT.F32 | Conversion to/from integer/fixed-point | 1 |
| VSQRT.F32 | Square root | 14 |

Las operaciones que vamos a realizar son la conversión de formato, productos y sumas, que realmente no consumen demasiada CPU, un ciclo cada una, pero en aritmética fija existen operaciones de producto y acumulación en un ciclo y no necesitamos convertir datos, quizá desplazar los resultados, así que probamos primero en coma fija. Lo que no tuvimos en cuenta fue el orden de magnitud de los factores que multiplican las entradas a filtrar y sus salidas anteriores, y en este punto nos vino muy bien la biblioteca empleada para las simulaciones. Estos coeficientes dependen de la frecuencia de muestreo, en nuestro caso varios MHz, y de la frecuencia de corte del filtro. Dando unos valores concretos, en el momento de realizar estas pruebas la frecuencia de muestreo estaba establecida en 4.5MHz, lo que es un tiempo de 2.2μs. Si queremos analizar una señal de 10kHz, que es un valor típico, ni muy alto ni muy bajo y filtrar a una frecuencia 10 veces menor para que la componente de $2\cdot\omega$ caiga un 99%, obtenemos coeficientes para el filtro del orden de $10^{-7}$ para las entradas y cercanos a la unidad para las salidas anteriores. Con unos cálculos sencillos, teniendo en cuenta que para cada 3 órdenes de magnitud se necesitan 10 bits, necesitaríamos más de 20 bits de parte decimal, por lo que las operaciones ya habría que realizarlas en anchos de palabra de 32 bits, con resultados de 64 y asignar la parte alta de la salida al acumulador. Quizá en este caso ya no esté tan justificado el uso de la coma fija, si la flotante nos lo da resuelto y las conversiones solo tardan un ciclo. El problema es que aún en representación flotante de precisión simple, la mantisa son 24 bits, que es más o menos la diferencia de magnitud de los coeficientes para esta configuración, sin ser la más extrema. Podría interesar realizar las operaciones en doble precisión, con el siguiente coste en ciclos obtenido de la misma nota de aplicación:

**Tabla 5-14. Ciclos necesarios para ejecutar algunas operaciones en doble precisión en M4/M7**

| Instruction | Description | Cycles |
|---|---|---|
| VADD.F64 | Addition | 3 |
| VSUB.F64 | Subtraction | 3 |
| VCVT.F<32\|64> | Conversion to/from Integer/fixed-point | 3 |

Aquí vemos que el coste en ciclos es bastante mayor, por lo que interesa limitar este tipo de operaciones, aunque se realicen en el núcleo más rápido. Una de las maneras es emplear una discretización del filtro analógico barata computacionalmente. En lugar de aplicar la transformación bilineal, más habitual cuando se busca precisión, empleamos el método de Euler implícito, que proporciona una fórmula más sencilla. Si el periodo de muestreo es suficientemente pequeño, como lo será en la mayoría de los casos, la diferencia va a ser despreciable. Y si no lo es, los propios efectos de la discretización por sí solos harán que el resultado no sea óptimo, por lo que tampoco estamos perdiendo demasiado, a cambio de obtener un código más eficiente. La transformación aplicada en el método de Euler implícito es, con T el periodo de muestreo:

$$s = \frac{z-1}{z\cdot T} = \frac{1-z^{-1}}{T}$$

El proceso sería el siguiente, partiendo de la función de transferencia continua:

$$H(s) = \frac{1}{\frac{1}{\omega_c^2}s^2 + \frac{\sqrt{2}}{\omega_c}s + 1} = \frac{y}{u}$$

$$u = \frac{y}{\omega_c^2}s^2 + \frac{y \cdot \sqrt{2}}{\omega_c}s + y$$

Aplicando la transformación al dominio z obtenemos:

$$\mathrm{u} = \frac{y \cdot (1 - 2 \cdot \mathrm{z}^{-1} + \mathrm{z}^{-2})}{T^2 \cdot \omega_c^2} + \frac{\mathrm{y} \cdot \left(\sqrt{2} - \sqrt{2} \cdot z^{-1}\right)}{T \cdot \omega_c} + \mathrm{y}$$

$$u_n \cdot T^2 \cdot \omega_c^2 = y_n(1 + \sqrt{2} \cdot T \cdot \omega_c + T^2 \cdot \omega_c^2) - y_{n-1}(2 + \sqrt{2} \cdot T \cdot \omega_c) + y_{n-2}$$

$$y_n = \frac{\left(2 + \sqrt{2} \cdot T \cdot \omega_c\right) \cdot y_{n-1} - y_{n-2} + T^2 \cdot \omega_c^2 \cdot u_n}{1 + \sqrt{2} \cdot T \cdot \omega_c + T^2 \cdot \omega_c^2}$$

En los casos en los que la frecuencia de corte del filtro sea muy inferior a la de muestreo, el producto $T \cdot \omega_c$ también lo será, pero los demás coeficientes serán del orden de las unidades. Esta circunstancia es la que hacía necesario emplear aritmética de coma flotante en doble precisión para evitar problemas con los cálculos. Con esta expresión solo hace falta realizar un producto en doble precisión, aparte de usarla para el acumulador, o lo que es lo mismo, la salida del filtro. Aun así, puede dar problemas en casos de diferencias extremas entre las frecuencias de corte y de muestreo, pero en esas condiciones no tendría demasiado sentido emplear todas las muestras que se generan, ya que es de esperar que de una a la siguiente haya poca diferencia, salvo el ruido. Se ha adaptado el filtro con una etapa previa de diezmado que genera una muestra a partir del promedio de n de entrada, de manera que el periodo de muestreo efectivo sea n veces mayor y se reduzca este efecto.

Se han simulado los resultados del filtro original y este simplificado, añadiendo un nuevo componente a la librería de Modelica llamado Butter2LPb y modificando el ejemplo FiltroGenerador para filtrar la misma señal con las dos expresiones, obteniendo resultados muy parecidos que justifican este cambio.

La ecuación de partida es para un filtro de ganancia unitaria, si queremos cambiar este valor es suficiente con multiplicar el coeficiente de la entrada por la ganancia deseada. En el modelo matemático del amplificador vimos que el resultado de multiplicar la señal a analizar por un seno y un coseno y filtrarla para eliminar las altas frecuencias era la representación en coordenadas cartesianas de un vector de módulo la mitad de la amplitud de la señal original, por lo que basta con multiplicar el coeficiente de $u_n$ por 2 para que sean las de un vector de módulo la amplitud de la original. Aparte de esto, puede interesar aplicar otras ganancias sobre los resultados, por ejemplo, para ajustar

la salida del DAC de resultados a su fondo de escala. Hemos añadido también este parámetro a los cálculos.

### 5.4.5 Interfaz de control y de resultados

Para poder cambiar los parámetros del amplificador lock-in, leer sus resultados y comprobar el uso máximo del *buffer* de comunicación entre núcleos, se han añadido unas instrucciones que se enviarán a través del puerto USB empleando un puerto serie virtual. Aunque la comunicación se ha implementado en el núcleo M7, que no tiene requisitos estrictos de tiempo real, las funciones que manejan el puerto serie consumen cierto tiempo de CPU, sobre todo las que dan formato a las cadenas que se enviarán de vuelta desde el microcontrolador (sprintf). Corremos el riesgo de que durante el tiempo en que se están realizando tareas como la anterior se desborde el *buffer*, perdiendo la información de algunas muestras. Para reducir la probabilidad de desbordamiento se utilizan comunicaciones cortas sin seguir un estándar, como podría ser SCPI (*Standard Commands for Programmable Instruments*, comandos normalizados para instrumentos programables), muy empleado en dispositivos similares, y respuestas cortas desde el amplificador lock-in. La Tabla 5-15 muestra la configuración del puerto serie virtual y la Tabla 5-16 la lista de comandos implementados. Todos son de un único carácter, seguido en su caso por el valor del parámetro a modificar y terminados en el carácter de nueva línea (‘\n’, código ASCII 10). El sistema diferencia entre mayúsculas y minúsculas.

**Tabla 5-15. Configuración del puerto serie**

| Parámetro | Valor |
|---|---|
| **Bits de datos** | 8 |
| **Paridad** | No |
| **Bits de parada** | 1 |
| **Velocidad (bps)** | 250.000 |

**Tabla 5-16. Instrucciones de control por el puerto serie**

| Código | Parámetro | Descripción |
|---|---|---|
| ? | No tiene | Solicita los valores X e Y actuales |
| ! | No tiene | Solicita el número máximo de muestras sin procesar que ha contenido el buffer desde que se ejecutó este comando antes |
| a | Número real | Establece la amplitud de excitación en V |
| f | Número real | Establece la frecuencia de trabajo en Hz |
| p | Número real | Establece la fase de referencia en rad |
| g | [0,1,2,3] | Establece la ganancia de entrada en $10^{\text{parámetro}}$ |
| G | Número real | Establece la ganancia de salida (adimensional) |
| c | Número real | Estable la frecuencia de corte del filtro IIR |

Los resultados de los cálculos se pueden obtener con la instrucción ‘?’ o analógicamente. En esta primera versión del firmware las salidas de resultados son fijas, mostrando los valores de X, Y y R de izquierda a derecha en los conectores.

### 5.4.6 Ajuste de la frecuencia de ciclo

Una vez completado el resto de la implementación probamos distintas configuraciones para sincronizar los convertidores analógicos y obtener tiempos de muestreo pequeños y reproducibles. La primera opción fue generar una señal de referencia TTL con un ciclo de trabajo del 50%, usarla como señal de reloj del ADC y emplear alguna de sus transiciones de nivel como sincronización para el ciclo del amplificador lock-in, de manera que se actualizara el DAC a partir de este flanco, se leyeran los datos del bus paralelo del ADC y el resto de las operaciones necesarias en cada ciclo en tiempo real. Esta sincronización se puede realizar mediante interrupciones o comprobando continuamente el estado de la señal (*polling*). Ya hemos justificado la conveniencia de deshabilitar las interrupciones en el núcleo encargado de las tareas en tiempo real y las medidas indican que se tarda solo un ciclo en leer una entrada digital accediendo directamente al registro adecuado, por lo que el *polling* parecía una buena idea. Midiendo el tiempo de ciclo, que según la carga de trabajo asignada a cada núcleo variaba entre unos 140ns y 180ns, procedimos a configurar el periodo de la señal PWM con estos valores o algunos poco superiores, obteniendo resultados incorrectos en la salida analógica del DAC. Inspeccionando la señal de selección del DAC, que esperábamos funcionase a la misma frecuencia que la del ADC, observamos que aproximadamente en uno de cada 5 o 6 ciclos permanecía sin cambios, lo que nos hizo plantearnos que el mecanismo de sincronización añadiera más tiempo del deseado, al menos en algunos casos. En efecto, la lectura de la señal no añadía mucha carga, pero sí lo hacía la comprobación de la condición de sincronización y la bifurcación según se cumpliera o no. Había que aumentar el tiempo de ciclo en unos 30ns para evitar estos efectos en el caso peor. Esto equivale a unos 6 u 8 ciclos, que no son muchos, pero suponen cerca del 20% de pérdida de eficiencia. En lugar de eso decidimos generar la señal de reloj del ADC actualizando una salida digital desde el propio ciclo a la vez que se actualiza la del DAC y medimos los nuevos tiempos del periodo en base a esta señal. Es importante tener en cuenta que no solo buscamos tiempos pequeños que permitan definir ciclos rápidos, sino que además necesitemos que todos duren aproximadamente lo mismo para que el muestreo sea homogéneo en el tiempo. Analizando la señal en un osciloscopio (Figura 43) apreciamos que el tiempo medio es muy pequeño (143.1ns) y con muy poca variación entre ciclos, que es lo que buscamos. Este tiempo equivale a 34 ciclos del reloj a 240MHz y una frecuencia de muestreo algo superior a 7MHz, un valor mayor que el que esperábamos conseguir en las estimaciones iniciales. Las oscilaciones de alta frecuencia se pueden deber a lo rudimentario de la medida y en cualquier caso no afectan al correcto funcionamiento, ya que no suponen un cambio en el valor lógico de la señal.

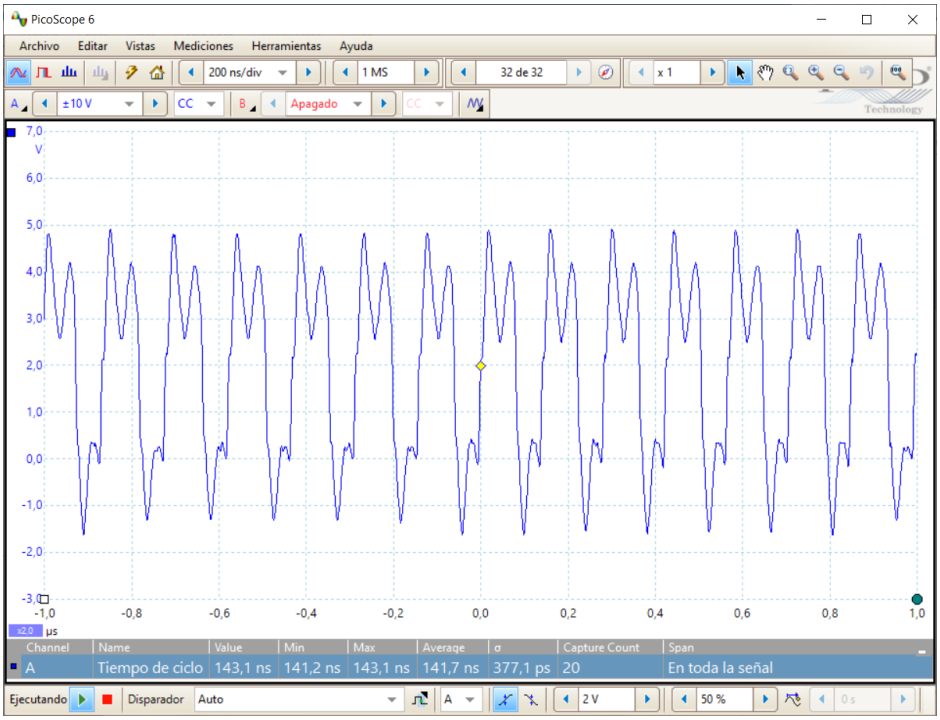



**Figura 43. Señal de sincronización para los convertidores analógicos**

## 5.5 Costes

Al elegir los componentes indicamos sus características destacadas y precios aproximados de algunos de ellos, justificando las decisiones en base a ambos aspectos. En este apartado presentamos el coste del material y ensamblado automático del prototipo desarrollado. Téngase en cuenta que solo se indican estos costes, el del material necesario para poder poner en marcha una unidad y el montaje de los componentes que se han pedido soldados a la placa. No se incluyen por tanto los costes de otros conceptos, como sería el desarrollo, el montaje de los componentes de orificio pasante, el ensamblado del conjunto, las pruebas, etc.

La empresa de fabricación y ensamblado de las placas de circuito impreso exige un pedido mínimo de dos unidades, por lo que se encargaron dos placas parcialmente montadas y los cálculos que se recogen en la Tabla 5-17 se han realizado en base a esta cantidad. Los costes que se indican son los reales para este volumen de producción en el momento de fabricar el primer prototipo. En la fecha de la escritura de este documento estos precios han variado, siendo en general menores, como también ha variado la disponibilidad de los componentes, habiéndose detectado que la referencia DAC81404 no se encuentra disponible en los distribuidores habituales y su plazo de reposición es de aproximadamente un año, lo que haría recomendable el rediseño para emplear un componente alternativo.

Otro aspecto que suele afectar al precio final de fabricación son los descuentos por volumen. En la fabricación y montaje de PCB es habitual que el precio de un lote de 5 placas cueste parecido a otro de 10, 20 o incluso 50. Esto se debe a que, sobre todo

en el ensamblado, el precio que se paga se debe principalmente a los servicios de ingeniería y preparación de las máquinas más que a su tiempo de uso, que realmente es pequeño en este tipo de pedidos. Incluso en lotes tan pequeños el precio del material en placas de pocas capas y tamaños reducidos no supone un gasto comparable a los costes fijos. Téngase en cuenta que el producto desarrollado no tendrá grandes tiradas, al ser de uso muy específico, por lo que no tiene sentido hablar de producciones de miles de unidades o cantidades superiores, como sucede con la electrónica de consumo, cuyo análisis de escalabilidad de precios sería muy distinto. Por ejemplo, en los volúmenes que nos ocupan el precio de los componentes casi no varía según la cantidad. Teniendo en cuenta lo reducido de los costes fijos en la empresa elegida, se puede considerar que el precio aquí calculado para una unidad sufrirá muy pocas variaciones si se fabrican lotes pequeños, ya que los costes a repartir suponen menos del 15% del total.

**Tabla 5-17. Coste de material del prototipo**

| Componente | Cantidad | Precio unitario | Total 2uds |
|---|---|---|---|
| Conector BNC acodado | 5 | 2,3150 € | 23,150 € |
| Cap 10uF | 9 | 0,1353 € | 2,435 € |
| Cap 100nF | 28 | 0,0028 € | 0,157 € |
| Cap 1uF | 2 | 0,0495 € | 0,198 € |
| Jumper_Header 2x1 | 4 | 0,0200 € | 0,160 € |
| Socket 10x2_N144_ARD | 1 | 0,2000 € | 0,400 € |
| Socket 8x2_N144_ARD | 1 | 0,0160 € | 0,032 € |
| Socket 15x2_N144_ARD | 1 | 0,3000 € | 0,600 € |
| Socket 17x2_N144_ARD | 1 | 0,3400 € | 0,680 € |
| 1N4148WS T4 | 2 | 0,0837 € | 0,335 € |
| CJ431 | 2 | 0,1866 € | 0,746 € |
| LED 0603 Green | 2 | 0,1523 € | 0,609 € |
| Caja metálica 1455T1602 | 1 | 27,8400 € | 55,680 € |
| Shunt Black 2.54mm | 4 | 0,0085 € | 0,068 € |
| CON DIN 5 SDS-50J | 1 | 1,4000 € | 2,800 € |
| Header 3 | 3 | 0,0300 € | 0,180 € |
| GZ2012D101TF | 3 | 0,0330 € | 0,198 € |
| SLP3-150-150-F | 2 | 0,4100 € | 1,640 € |
| AC-C13 EU | 1 | 4,8600 € | 9,720 € |
| GP25A13D-R1B | 1 | 40,7800 € | 81,560 € |
| SS8050 Y1 | 2 | 0,0072 € | 0,029 € |
| Res 10K | 2 | 0,0064 € | 0,026 € |
| Res 33 | 4 | 0,0034 € | 0,027 € |
| Res 20K | 2 | 0,0073 € | 0,029 € |
| Res 4.99K | 2 | 0,0064 € | 0,026 € |
| Res 100K | 1 | 0,0118 € | 0,024 € |
| Res 33K | 1 | 0,0127 € | 0,025 € |
| Res 1K | 6 | 0,0038 € | 0,046 € |

| Componente | Cantidad | Precio unitario | Total 2uds |
|---|---|---|---|
| Res 100R | 1 | 0,0284 € | 0,057 € |
| Res 16K | 2 | 0,0137 € | 0,055 € |
| Res 10K | 5 | 0,0047 € | 0,047 € |
| Res 2.2K | 2 | 0,0082 € | 0,033 € |
| Res 1M | 1 | 0,0245 € | 0,049 € |
| Res 0 | 2 | 0,0118 € | 0,047 € |
| Res 40.2K | 1 | 0,0490 € | 0,098 € |
| Res 2K | 1 | 0,0264 € | 0,053 € |
| G6K-2F-Y-TR DC12 | 2 | 4,4650 € | 17,860 € |
| DAC81404RHBT | 1 | 33,8700 € | 67,740 € |
| AD5447YRUZ | 1 | 31,3432 € | 62,686 € |
| OPA2197IDR | 4 | 2,2920 € | 18,336 € |
| AD9240ASZ | 1 | 24,3454 € | 48,691 € |
| TXB0104PWRG4 | 1 | 0,9043 € | 1,809 € |
| Micro USB cable | 1 | 3,7400 € | 7,480 € |
| Panel frontal | 1 | 1,3380 € | 2,676 € |
| Panel trasero | 1 | 1,3380 € | 2,676 € |
| PCB 4 capas 160x160 | 1 | 10,7527 € | 21,505 € |

| | |
|---|---|
| Total para 2 unidades | 433,48 € |

A este coste de material, que incluye las placas de circuito impreso, hay que añadirle el de ensamblado y transporte. La Tabla 5-18 incluye estos conceptos junto con el coste de los componentes y muestra el precio de fabricación unitario del prototipo.

**Tabla 5-18. Costes adicionales de fabricación**

| Concepto | Precio |
|---|---|
| Componentes | 433,48 € |
| Servicio de montaje | 20,47 € |
| Transporte | 16,53 € |

| | |
|---|---|
| Total fabricación para 2 unidades | 470,48 € |
| Total fabricación para 1 unidad | 235,24 € |

# CAPÍTULO 6. VALIDACIÓN

En este capítulo vamos a presentar los resultados de la validación del prototipo, sus características finales, el programa de control empleado para comprobar el correcto funcionamiento y los resultados obtenidos con él.

## 6.1 Configuración, correcciones y mejoras sobre el diseño e implementación

### 6.1.1 Configuración *hardware*

Cuando realizamos el diseño de la placa partimos de un conjunto de circuitos integrados, sus componentes auxiliares y otros elementos. Era la primera vez que usábamos muchos de ellos, o la primera vez que empleábamos ciertas configuraciones. Cuando esto sucede y el diseño tiene cierta complejidad es habitual emplear módulos de evaluación de los componentes más críticos, como podrían ser los DAC y ADC, para validar sus características, interfaces de comunicación, configuración, etc. En nuestro caso, por abaratar costes y reducir tiempos de desarrollo, nos hemos saltado este proceso y hemos diseñado en base a las hojas de características. Si se siguen todas las indicaciones de los distintos fabricantes el diseño debe funcionar bien, pero en general es más fácil cometer algún error o no terminar de entender algún dato de cómo configurar o manejar el componente. Como medida preventiva ante estas y otras posibles eventualidades se implementaron varias alternativas en funcionalidades que, o no estaban muy claras según la información técnica, o parecían más complejas y con mayor probabilidad de fallo en la implementación real. Estas alternativas se pueden elegir mediante un conjunto de puentes que se colocan en la placa y su configuración final se detalla en la Tabla 6-1, donde se indica qué pines del conector se deben unir entre ellos.

**Tabla 6-1. Configuración de los puentes**

| Conector | Pin 1 | Pin 2 | Pin 3 | Función |
|---|---|---|---|---|
| **JP1** | ■ | ■ | | Amplitud controlada con DAC8104 |
| **JP2** | ■ | ■ | | Señal de reloj del ADC en pin PD15 |
| **JP3** | | ■ | ■ | Señal de reloj del ADC a nivel LVTTL |

La Figura 44 muestra el detalle de estos puentes en la placa.

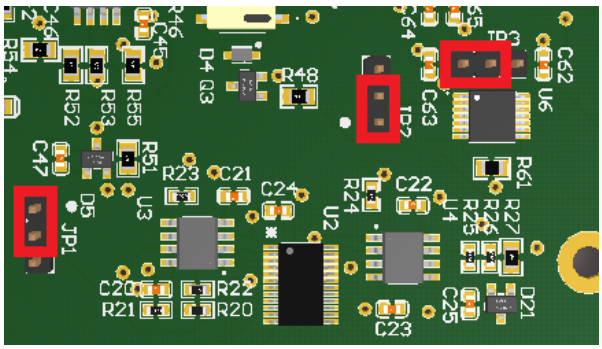



**Figura 44. Configuración de los puentes**

### 6.1.2 Modificaciones a los esquemáticos

Durante la etapa de validación se han detectado una serie de errores o mejoras en el diseño de los esquemáticos y posterior implementación del circuito electrónico, que se detallan a continuación y se deben tener en cuenta en el futuro si se fabrican más unidades de este mismo prototipo o se decide emplear como base para uno mejorado. También se incluyen modificaciones para eliminar los componentes que no se usan por haberse añadido varias opciones, tal y como se ha indicado en el epígrafe 6.1.1. Algunas de estas correcciones se han implementado manualmente en la unidad montada, desoldando los componentes sobrantes, cortando las pistas afectadas y soldando cables para unir los elementos necesarios.

#### 6.1.2.1 Alimentación del DAC AD5447

Se ha detectado un error en el pin 21 del componente U2 (VDD). En lugar de emplear los 5V analógicos (5V_A) se deben emplear los 3.3V digitales (3V3).

#### 6.1.2.2 Filtrado de las referencias de tensión

Junto con las referencias de tensión de 2.5V (CJ431) se incluyeron condensadores en paralelo para reducir el ruido, consiguiendo el efecto contrario. Se deben eliminar los componentes C25 y C47.

#### 6.1.2.3 Señal de activación del convertidor de nivel TXB0104

Aunque su uso no es necesario y se propone su eliminación en la modificación 6.1.2.5, si se desease emplear el nivel TTL para la señal de reloj del ADC sería necesario unir el pin 8 de U6 con la señal de alimentación de 3.3V.

#### 6.1.2.4 Selección de la señal de reloj para el ADC

La opción elegida para la generación del reloj del ADC AD9240 hace innecesario mantener la posibilidad de configurarlo manualmente, por lo que se puede eliminar el componente JP2 y unir sus pines 1 y 2.

#### 6.1.2.5 Selección del nivel de tensión del reloj del ADC

Una vez comprobado que el ADC AD9240 funciona correctamente con nivel de tensión LVTTL en la señal de reloj, se propone eliminar los componentes encargados del cambio de nivel a TTL y la selección de la señal a emplear. Esto incluye los componentes U6, JP3, R61, C62 y C63. El pin 7 de U5 se debe conectar directamente al puerto CLK.

#### 6.1.2.6 Selección de la señal de control de la amplitud de excitación

Inicialmente se planteó una alternativa al control de la amplitud de la señal de excitación por si surgían problemas con el manejo del DAC DAC81404. Dichos problemas no aparecieron, por lo que se pueden eliminar esos componentes y unir el puerto VREF al pin 3 del componente U2. Los elementos que se pueden eliminar del diseño son U4, JP1, C22, C23, C25, F21, R24, R25, R26 y R27.

### 6.2 Aplicación de control y validación

Para facilitar las pruebas de las distintas partes que componen en amplificador lock-in y poder leer los resultados de los cálculos digitalmente, sin pérdida de calidad por su conversión posterior a tensión, hemos implementado una aplicación en Matlab con una interfaz gráfica. Permite modificar los parámetros de funcionamiento, leer los resultados de los cálculos e incluso representar gráficamente las señales X, Y, R y $\theta$ en función de la frecuencia, realizando barridos en los que este parámetro evolucione de manera lineal (aumentando la misa cantidad en cada paso) o exponencial (aumentando la misma proporción en cada paso). Hemos elegido el inglés como idioma para la interfaz, al igual que para el código fuente, pensando en un posible uso real en laboratorios de física o ingeniería, donde este suele ser el idioma empleado internacionalmente.

La Figura 45 muestra una captura de pantalla de la aplicación e incluye un ejemplo de las curvas citadas. Los datos corresponden a medidas reales sobre un circuito RLC, en el que se aprecia perfectamente la ganancia máxima en la frecuencia de resonancia, los valores medidos en el entorno de este valor y la fase, nula en la resonancia y cercana a $90^{o}$ y $-90^{o}$ lejos de ella. Dado que la amplitud de excitación es 5V, la ganancia en módulo del sistema equivaldría al valor de la línea amarilla dividido entre estos 5V. La Figura 46 muestra el resultado de la simulación en Altium, donde aparte del circuito RLC propiamente dicho se ha incluido la impedancia de salida del amplificador operacional y la resistencia parásita de la bobina. Este par de imágenes se puede considerar la prueba de que el dispositivo funciona correctamente, al menos en este intervalo de frecuencias.

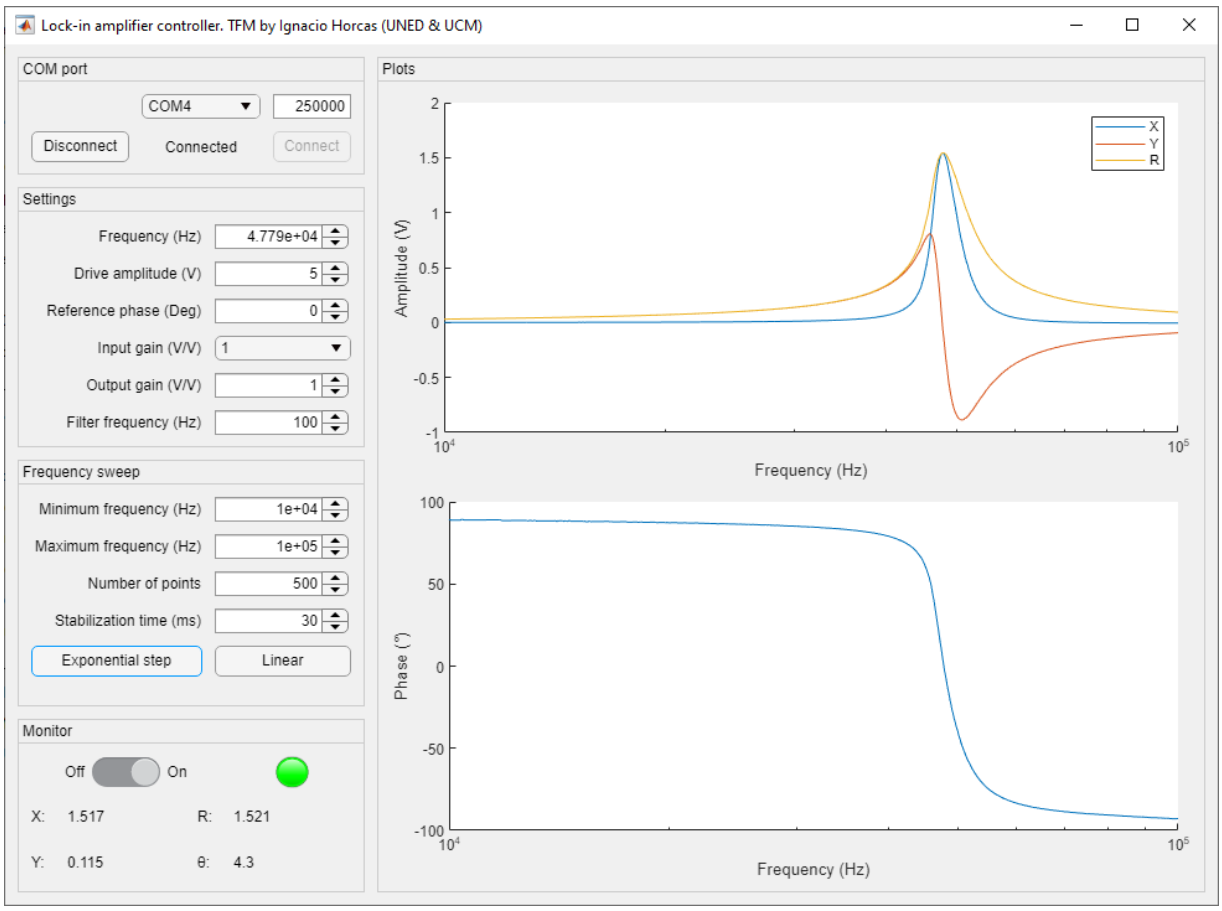



**Figura 45. Interfaz de usuario de la aplicación de control y validación desarrollada**

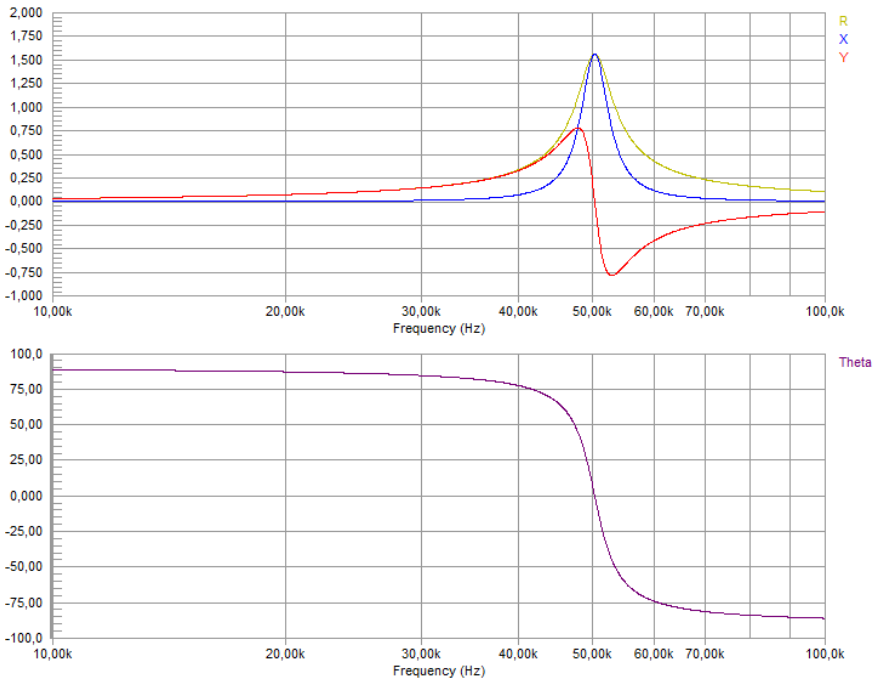



**Figura 46. Simulación del circuito RLC empleado**

### 6.2.1 Interfaz de usuario

Los elementos de la interfaz gráfica de usuario están agrupados en 4 bloques distintos según su funcionalidad. Como parte de la gestión de la interfaz se ha añadido una máquina de estados que deshabilita los controles o bloques cuando no tenga sentido su uso y los vuelve a habilitar cuando sí. Por ejemplo, hasta que no esté conectado e inicializado el amplificador lock-in no tiene sentido leer los resultados de la medida y este grupo estará deshabilitado.

#### 6.2.1.1 COM port

Contiene la configuración del puerto serie que se empleará para comunicarse con el dispositivo, tanto para configurarlo como para recibir los resultados digitalmente. Una vez configurado el número de puerto y su velocidad, que debe ser la de la Tabla 5-15, se debe pulsar el botón “Connect”, que abre el puerto y manda los parámetros de configuración al amplificador lock-in. El botón “Disconnect” cierra el puerto y permite cambiar la configuración para seleccionar otro distinto u otra velocidad de comunicación.

#### 6.2.1.2 Settings

Permite cambiar los parámetros de funcionamiento del amplificador lock-in. En el momento de terminar la edición de un campo, bien sea porque se valida el dato o porque se cambia a otro elemento de la interfaz, se envía el dato de configuración al dispositivo. Cada parámetro de la interfaz tiene relación directa con alguno de los ya vistos durante esta memoria. Esta relación se recoge en la Tabla 6-2.

**Tabla 6-2. Parámetros de control de la interfaz de usuario**

| Parámetro | Unidad | Significado |
|---|---|---|
| Frequency | Hz | Frecuencia de la señal de excitación generada y de la señal a analizar |
| Drive amplitude | V | Amplitud de la señal de excitación |
| Reference phase | º | Desfase de la señal de referencia considerada el origen de fases |
| Input gain | V/V | Ganancia analógica de la etapa de entrada. Sus valores son discretos {1, 10, 100, 1000} |
| Output gain | V/V | Ganancia digital aplicada a las señales de resultados {X, Y, R} |
| Filter frequency | Hz | Frecuencia de corte del filtro digital pasa baja que se aplica a las señales X e Y |

#### 6.2.1.3 Frequency sweep

Permite realizar barridos en frecuencia, esto es, variar la frecuencia de las señales generada y analizada entre dos límites mientras se miden los resultados en cada una de estas configuraciones. Es útil para caracterizar el elemento que se esté probando, que

puede ser el propio amplificador lock-in. La Tabla 6-2 recoge el significado de cada elemento.

**Tabla 6-3. Parámetros del barrido de frecuencias en la interfaz de usuario**

| Parámetro | Unidad | Significado |
|---|---|---|
| Minimum frequency | Hz | Valor inicial del barrido de frecuencia |
| Minimum frequency | Hz | Valor final del barrido de frecuencia |
| Number of points | | Número de frecuencias distintas que se usarán para el barrido |
| Stabilization time | ms | Tiempo de espera desde que se establece un nuevo de valor de frecuencia hasta que se considera que el sistema está estable y se miden los resultados |
| Exponential step | | Realiza el barrido variando la frecuencia de manera exponencial, es decir, la frecuencia en un determinado punto equivale a la del anterior multiplicada por un factor constante. También permite detener el barrido mientras está activo |
| Linear | | Realiza el barrido variando la frecuencia de manera lineal, es decir, la frecuencia en un determinado punto equivale a la del anterior más un valor constante. También permite detener el barrido mientras está activo |

#### 6.2.1.4 Monitor

Contiene un pulsador que activa la monitorización digital de los resultados, de manera que aparte de generarse las correspondientes señales analógicas en los conectores de resultados se muestren en la parte inferior de este grupo de controles. El indicador permanecerá de color verde mientras esté activo y gris cuando no lo esté.

#### 6.2.1.5 Plots

Muestra los resultados del barrido de frecuencia, empleando escala logarítmica para el eje de frecuencias y lineal para el de resultados. La curva superior representa los valores de las señales X, Y y R como amplitud de oscilación y la inferior los de $\theta$. Para las normalizaciones, tanto de estos resultados como los que se muestran en el grupo "Monitor" se tienen en cuenta tanto las ganancias de entrada como las de salida, a diferencia de las adaptaciones que se plantearon en el apartado 3.2.3, donde se indicó que estas ganancias se podían compensar. Cada una de las opciones tiene sus ventajas e inconvenientes. En este caso hemos preferido no compensar las normalizaciones para obtener los mismos resultados que en las salidas analógicas, que siempre deben ir sin compensar para aprovechar el fondo de escala de los DAC de resultados y así mejorar la relación señal/ruido. También nos permite usar la ganancia de salida para compensar libremente otros factores y emplear nuestras propias normalizaciones, algo de lo que nos beneficiaremos en las pruebas del ancho de banda para mostrar curvas de ganancia de

la función de transferencia para distintas amplitudes de trabajo y ganancias, en lugar de curvas de la amplitud medida.

### 6.3 Caracterización del ruido de la entrada

Al definir las especificaciones mínimas a alcanzar y las ideales presentamos una breve discusión sobre distintas características del ruido analógico que podía ser deseable conocer. En este apartado vamos a presentar una caracterización sencilla del ruido de la etapa analógica de entrada, ya que un estudio completo, aunque sin duda es muy interesante y necesario si se quisiera comercializar este desarrollo, consideramos que queda fuera del alcance de este TFM por su extensión y los medios técnicos necesarios para llevarlo a cabo.

La primera prueba consiste en medir la relación de rechazo al ruido, para lo que empleamos un generador de funciones analógico y conectamos su salida a la entrada del amplificador lock-in mientras genera un seno de frecuencia fija. Idealmente deberíamos obtener un pico de la amplitud generada en su frecuencia y valores nulos en el resto de las frecuencias. La Figura 47 muestra el resultado de este análisis, donde se ha normalizado la amplitud medida empleando la ganancia digital y se han expresado los valore en dB, unidad logarítmica más habitual en este tipo de análisis. Aparte del pico, con un cierto ancho, se aprecian otras componentes de mayor frecuencia que probablemente se deban a armónicos de la señal de entrada. Téngase en cuenta que el generador de funciones empleado es de características básicas y que la magnitud de estas componentes es inferior a -40dB, lo que equivale a valores menores del 1%. Aparte de estos picos y la zona de la frecuencia a analizar, los valores medidos en general se mantienen por debajo de -80dB, un valor muy bueno, cercano al límite que establece la propia conversión a digital usando un ADC de 14 bits, donde la relación entre el fondo de escala y el valor del bit menos significativo es 84.3dB. Las medidas están realizadas con la frecuencia de corte del filtro establecida a 1Hz.

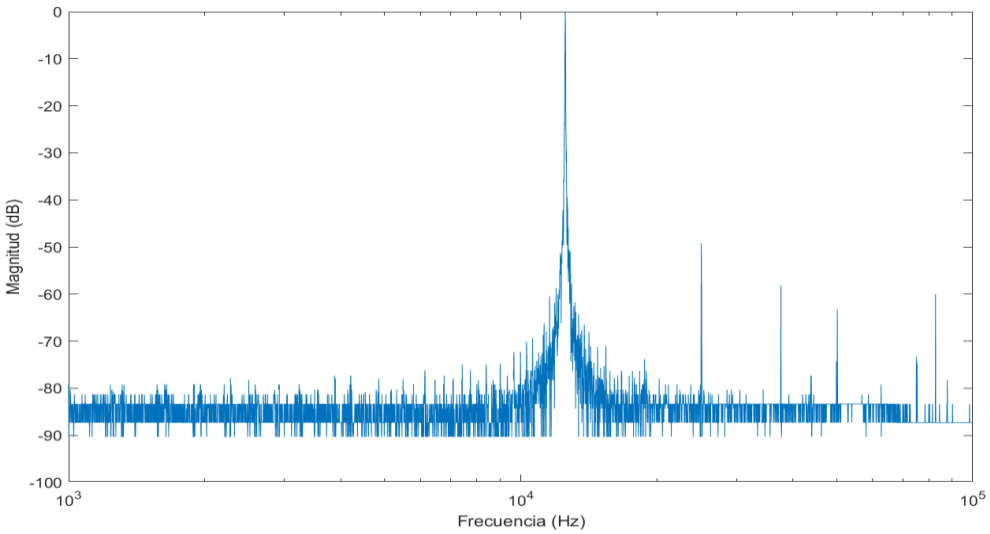



**Figura 47. Rechazo al ruido ante una entrada sinusoidal de frecuencia fija**

La siguiente prueba ha consistido en poner a 0V la entrada analógica y analizar la señal leída en el ADC, lo que nos dará una idea del ruido de la etapa analógica para señales muy pequeñas. La manera de establecer este voltaje ha sido cortocircuitando la señal del conector BNC con su carcasa, que está puesta masa. Para mejorar la resolución de los resultados hemos aumentado al máximo la ganancia analógica de entrada, esto es, un factor 1000, y hemos establecido el mismo valor para la ganancia digital sobre los resultados, obteniendo una ganancia total de $10^6$V/V, lo que es equivalente a 1V/μV. Esto quiere decir que, aunque la Figura 48 exprese los datos en V, la unidad real de los valores medidos es μV. Podemos apreciar que para todo el rango de funcionamiento esperado (entre 100Hz y 2MHz) el valor de la amplitud (R) es inferior a 1uV, siendo un valor realmente bajo. De nuevo, las medidas están realizadas con 1Hz de frecuencia de corte del filtro.

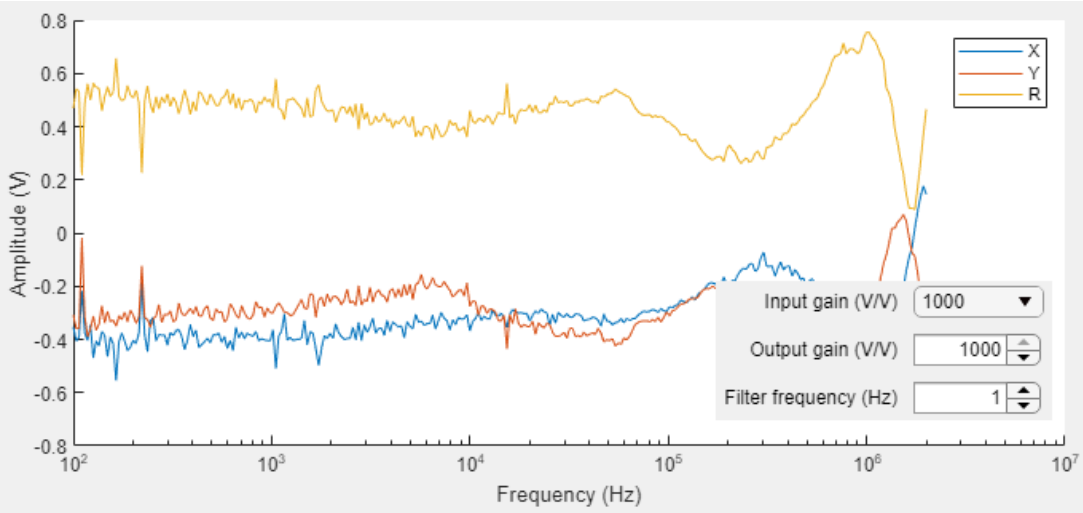



**Figura 48. Ruido con la entrada puesta a masa y ganancia total 1V/uV**

## 6.4 Pruebas de ancho de banda

La Figura 45 muestra el correcto funcionamiento para frecuencias entre 10kHz y 100kHz. Para probar valores fuera de este intervalo podríamos construir otros circuitos RLC con componentes que proporcionaran distintas frecuencias de resonancia o, más sencillo, conectar la señal de excitación a la entrada a analizar. El valor ideal de R en este caso es la amplitud de oscilación seleccionada. Para probar frecuencias pequeñas debemos configurar el filtro pasa-baja de los resultados también a valores pequeños, de manera que se elimine la componente de $2\cdot\omega$, lo que hace que la prueba sea relativamente lenta. Si realizamos un barrido de frecuencias entre 10Hz y 1kHz obtenemos el resultado de la Figura 49. En este caso, aunque se ha mantenido la amplitud de trabajo a 5V, se ha adaptado la ganancia de salida para que el valor teórico medido sea 1V y se facilite la interpretación de la curva.

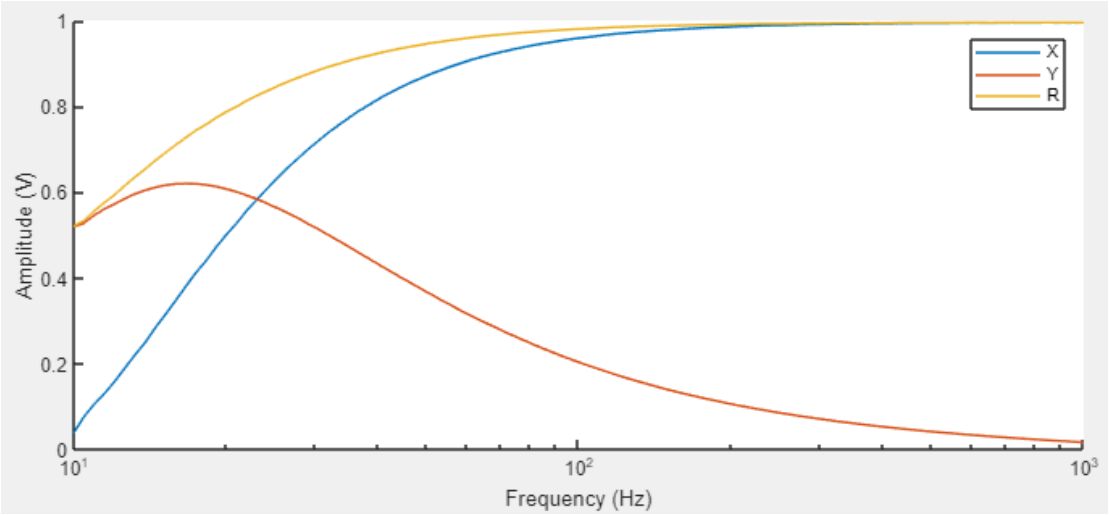



**Figura 49. Barrido a bajas frecuencias**

En la amplitud medida (línea amarilla) apreciamos que la ganancia a bajas frecuencias es menor que la unidad, lo que significa que el filtro pasa-alta a la entrada, que elimina la componente de continua, funciona. Si medimos la frecuencia en la que esta ganancia es -3dB obtenemos un valor de unos 16Hz, cuando lo habíamos diseñado para 10Hz. Esta diferencia realmente no supone un problema, se había elegido 10Hz como una frecuencia baja que eliminase la componente DC pero permitiera trabajar a valores más habituales, para lo que también es válido 16Hz y si no se considerase así se podrían reemplazar las resistencias y condensadores que configuran el filtro.

Si medimos la ganancia a frecuencias altas los resultados son distintos y dependen mucho de la configuración analógica. Las siguientes curvas (Figura 50 a Figura 53) muestran en amarillo la amplitud medida para distintas configuraciones de amplitud de excitación y ganancia analógica en barridos de frecuencia entre 100kHz y 2MHz. Recordemos que en el subtítulo 5.1, cuando definimos las especificaciones, habíamos establecido el ancho de banda ideal en 500kHz, por lo que estas pruebas superan ese límite, y que al elegir los componentes analógicos tuvimos en cuenta este valor e incluso el *slew rate* del amplificador operacional elegido podía ser un poco justo. Las propias figuras incluyen la configuración empleada en cada caso.

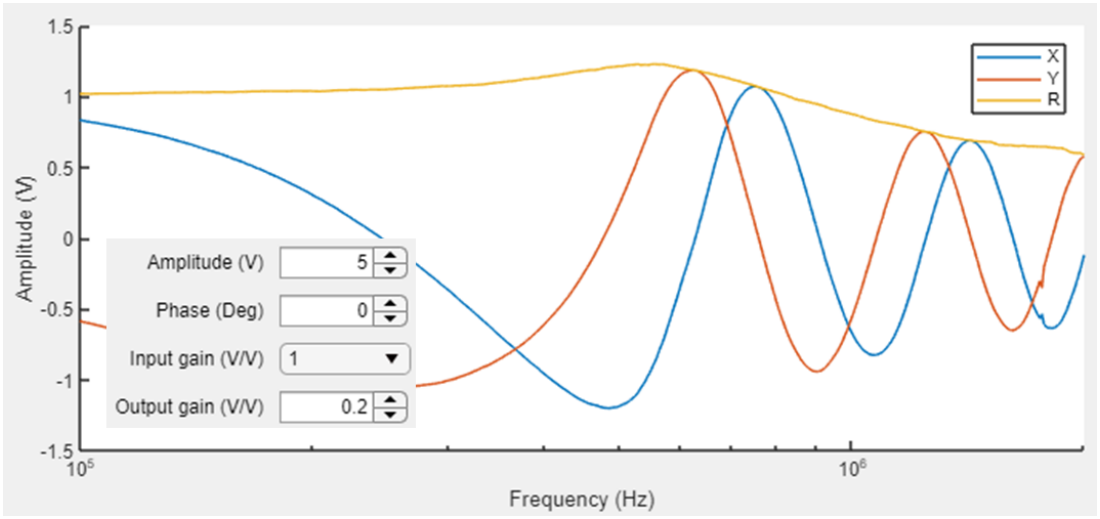



**Figura 50. Barrido entre 100kHz y 2MHz. Configuración 1**

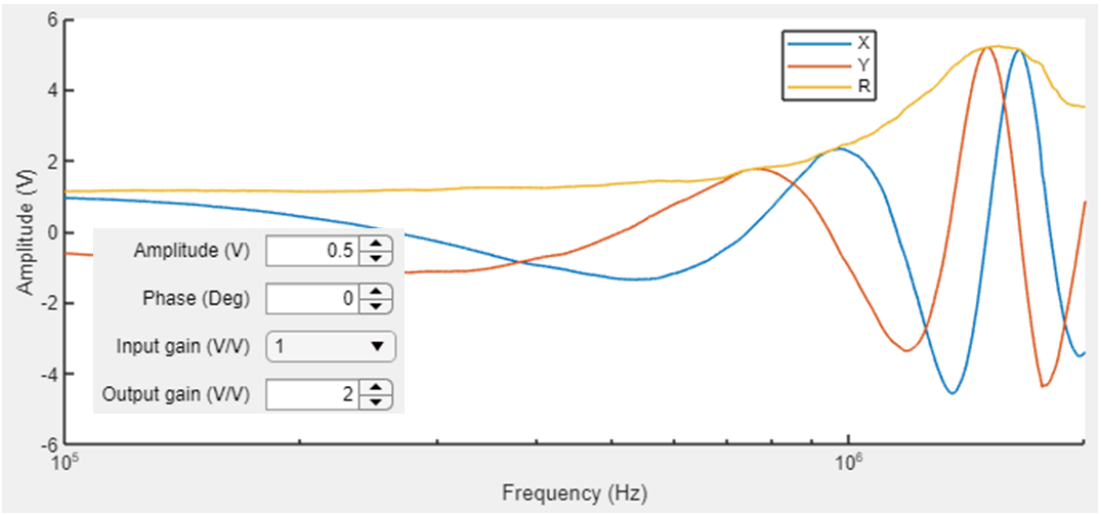



**Figura 51. Barrido entre 100kHz y 2MHz. Configuración 2**

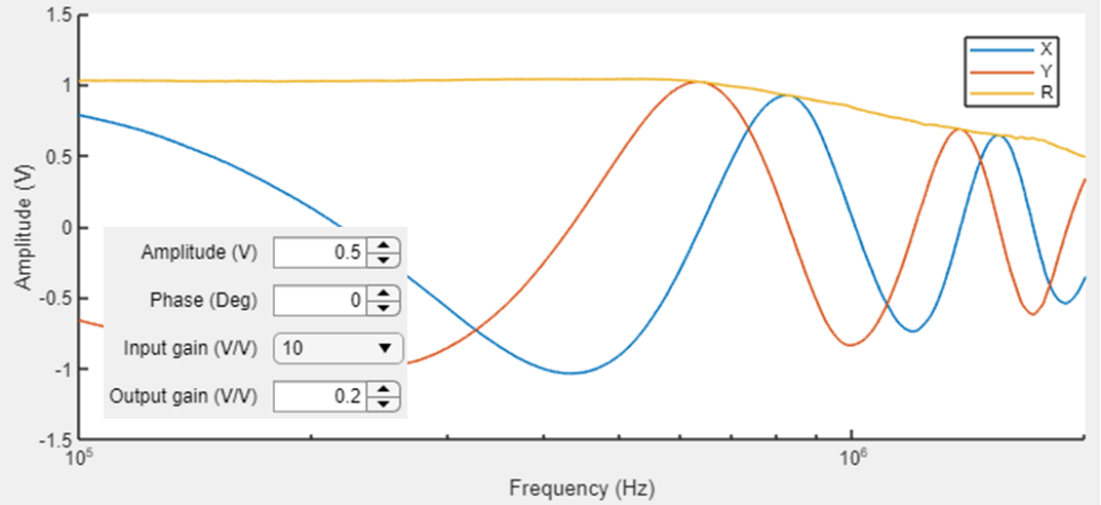



**Figura 52. Barrido entre 100kHz y 2MHz. Configuración 3**

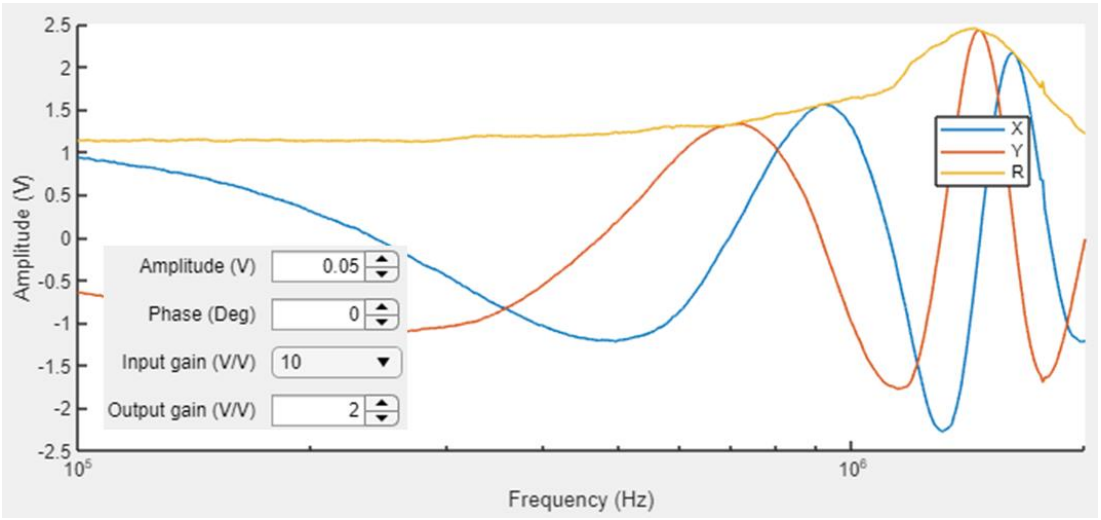



**Figura 53. Barrido entre 100kHz y 2MHz. Configuración 4**

En un primer análisis rápido de las 4 curvas en conjunto, fijándonos solo en las líneas amarillas (módulo de la función de transferencia), observamos que las formas coinciden bastante bien con determinados sistemas reales e incluso se aproximan a sistemas teóricos de segundo orden con mayor o menor coeficiente de amortiguamiento. Esto nos hace pensar que el procesado digital esté funcionando bien hasta estas frecuencias. De no ser así, las curvas no serían tan reconocibles. Si tenemos en cuenta nuestra frecuencia de muestreo (aproximadamente 7MHz) y comparamos en especificaciones de amplificadores comerciales este valor con la frecuencia máxima que dicen ser capaces de analizar (Tabla 2-1) vemos que en un caso el factor es 3 (1.8 GSPS y 600MHz) y en el otro prácticamente 2 (1.25GSPS y 600MHz). Aplicando el mismo criterio, con unas etapas analógicas ideales podríamos llegar a un ancho de banda de entre 2.3MHz y 3.5MHz, unos valores muy por encima de los esperados. También vemos que en la configuración 3 (Figura 52) la respuesta es prácticamente plana hasta unos 600kHz. Cabe esperar que reemplazando los amplificadores operaciones por otros de

mayor *slew-rate* y producto ganancia·ancho de banda consigamos mejores resultados, quizá llegando hasta varios MHz de frecuencias a analizar.

A partir de las curvas anteriores no podemos obtener información fiable sobre la fase de la señal en el rango de frecuencias medido. Apreciamos que las señales x e y varían de manera aparentemente arbitraria, generando ciclos que no deberían existir, ya que la fase lejos de los polos y los ceros de la función de transferencia se tiene que mantener constante y lo mismo debe suceder con la relación entre las señales x e y. A continuación analizamos el origen de este problema y la corrección añadida para solucionarlo.

## 6.5 Latencia de los convertidores

Analizando los valores de frecuencia donde se producen los distintos máximos y mínimos en las señales x e y de cada curva mostrada en el análisis de ancho de banda, apreciamos que aparecen a intervalos de frecuencia más o menos constantes, lo que podría indicar que la fase varía linealmente con la frecuencia. Para confirmarlo realizamos un barrido de frecuencia similar, pero en este caso representamos la fase en función de la frecuencia y usamos escalas lineales en lugar de logarítmicas, tal y como mostramos en la Figura 54, obteniendo una dependencia aproximadamente lineal con saltos al pasar de -180º a 180º.

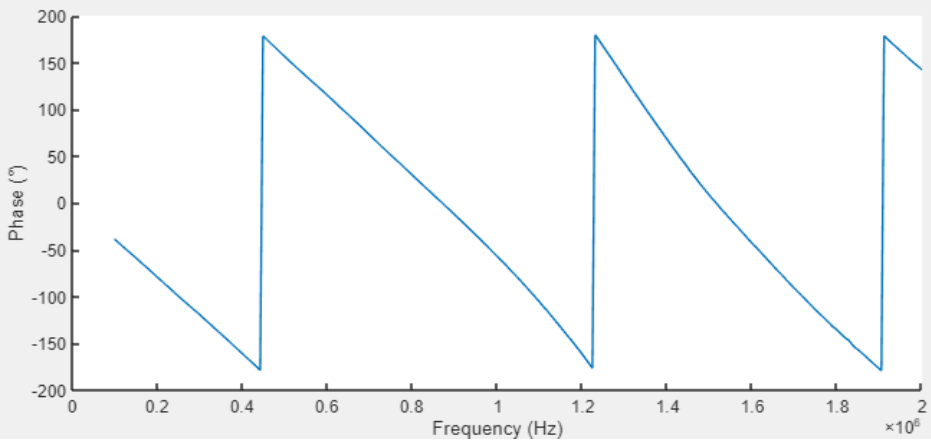



**Figura 54. Evolución de la fase en función de la frecuencia**

Esta dependencia casi lineal sugiere que en una primera aproximación el desfase entre la señal generada y la analizada sea un tiempo constante, más concretamente el inverso de la distancia en frecuencias entre los picos de la curva anterior.

En la simulación 4.4.7, en la que empleábamos el PLL para mantener el sistema en resonancia, indicamos sin entrar en mucho detalle que se compensaba la fase que introducía el sistema de medida. Recordemos también que las hipótesis de modelado suponían que no había retardos en los cálculos ni en la propagación de las señales, y aun así existía un desfase instrumental que compensar, algo que parecía contradictorio. Estos retardos en aquella ocasión se debían a los convertidores de analógico a digital y de digital a analógico. Si bien es cierto que se implementaron con un tiempo de

conversión nulo, ambos dispositivos tienen un periodo de retención antes del muestreo o mantienen la salida estable durante ciclos completos, lo que añade ciertos retardos del orden de su periodo de muestreo. En convertidores reales, sobre todo en ADC, suele haber retardos adicionales debidos a su arquitectura, típicamente segmentada (*pipelined*). La consecuencia es que el tiempo que tarda el dato en estar disponible aumenta y es un múltiplo entero del periodo de muestreo. A estos retardos se les suele llamar latencia (*latency*) y se miden en número de ciclos, en nuestro caso son 6. Al ser un sistema digital reprogramable es relativamente sencillo compensar este efecto. Basta con desfasar las señales I y Q y la analizada estos 6 ciclos antes de multiplicarlas. Implementando esta corrección el efecto desaparece, obteniendo curvas en las que el módulo y la fase se aproximan más a las típicas de los diagramas de Bode, como puede ser la Figura 55, medida en condiciones similares a las de la Figura 52 pero una vez implementada la corrección de la latencia.

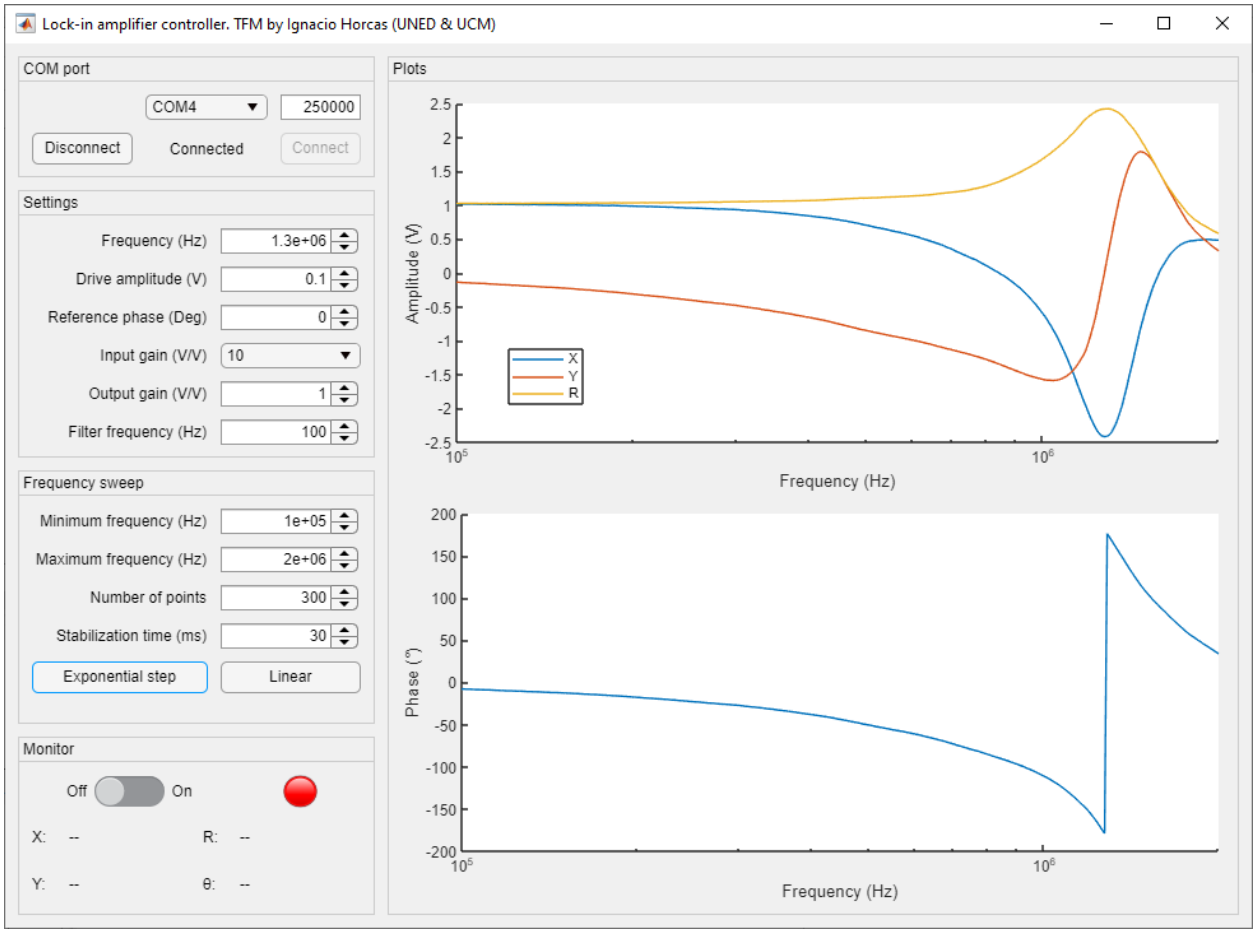



**Figura 55. Barrido de frecuencia con la latencia compensada**

## 6.6 Especificaciones finales

La Tabla 6-4 recoge la comparativa de especificaciones iniciales, con las dos opciones que habíamos planteado, y las finales, en las que vemos que las características se han cumplido o superado en todos los puntos.

**Tabla 6-4. Comparativa de especificaciones finales e iniciales**

| | Básico | Ideal | Conseguido |
|---|---|---|---|
| **Frecuencia de muestreo (kHz)** | 10 | 5000 | 7047 |
| **Ancho de banda (kHz)** | 1 | 500 | 600[5] |
| **Rechazo al ruido (dB)** | 40 | 60 | 80 |
| **Rango analógico (±V)** | 1 | 10 | 10 |
| **Resultados** | X, Y analógico (1Hz) | X, Y, R, θ, analógico (10kHz) y digital (USB/ethernet) | X, Y, R, analógico (40kHz) y digital (USB) |
| **Configuración** | Fija | USB/Ethernet | USB |

## 6.7 Validación como parte de un sistema de control real. Uso en un AFM

Una vez realizadas las pruebas unitarias de los distintos componentes y del amplificador completo, hemos querido verificar si realmente se puede utilizar como parte de un sistema de control, que es el objetivo final del instrumento y el motivo de que sea un TFM en ingeniería de sistemas y control. El sistema elegido ha sido un microscopio de fuerzas atómicas (AFM, *Atomic Force Microscope*) [26]. Las pruebas se han realizado en colaboración con el grupo Nanoforces de la Universidad Autónoma de Madrid [27], que durante muchos años ha desarrollado esta tecnología junta a la ya extinta empresa Nanotec Electrónica S.L. y desde su desaparición continúa desarrollando tanto la parte mecánica como la electrónica y el programa de control y procesado de datos WSxM [28].

### 6.7.1 Introducción a la microscopía de sonda de barrido

Los microscopios de fuerzas atómicas forman parte de la familia de los microscopios de sonda de barrido (*Scanning Probe Microscope*, SPM) y surgieron como evolución del microscopio de efecto túnel (*Scanning Tunnel Microscope*, STM) [29] por cuyo desarrollo sus inventores obtuvieron el premio Nobel de física tan solo 5 años después, en 1986 [30]. En todos los SPM una punta muy afilada recorre la superficie de la muestra midiendo alguna interacción física entre estos dos elementos y generalmente reaccionando ante esta medida. Por tanto, a diferencia de los microscopios ópticos, los SPM no emplean luz ni pueden distinguir colores, sino que crean mapas de alturas, corriente eléctrica, interacción magnética u otras magnitudes. Por comodidad se suele asignar una paleta de colores, que bien puede ser de escala de grises, representando cada tono un valor distinto, o con cualquier otra asignación de colores arbitraria. Lo habitual es que los colores oscuros representen valores menores que los claros, pero como decimos la asignación de colores a alturas (o a la magnitud representada) es arbitraria y no siempre tiene que ser así. En las imágenes que presentemos aquí sí se

[5] Según la configuración analógica

cumple que tonos más oscuros representan alturas menores y tonos más claros alturas mayores.

Para mantener constante la interacción entre la punta y la muestra se puede emplear cualquier tipo controlador, lo habitual es que sean controladores PI. Si suponemos que la magnitud medida solo depende de la distancia entre estos elementos y conseguimos mantenerla constante podemos decir que la distancia entre la punta y la muestra permanece constante durante toda la imagen. Recorriendo la muestra con la punta podemos obtener un mapa a una distancia fija, de manera que si vamos almacenando la altura de la punta en cada instante tendremos una copia de la superficie, desplazada una cierta cantidad constante. Estos movimientos, tanto en el eje Z para mantener constante la interacción, como en el plano XY para recorrer la superficie, se realizan empleando el efecto piezoeléctrico inverso. Los materiales piezoeléctricos son aquellos que sometidos a una fuerza producen una tensión eléctrica, que es aproximadamente proporcional a la fuerza recibida, y esta propiedad se emplea a menudo en distintos tipos de sensores de fuerza o de presión. El efecto es reversible, si se somete el material a una diferencia de potencial entre sus extremos se deforma de manera aproximadamente proporcional a la tensión aplicada, estirándose o encogiéndose. Los actuadores piezoeléctricos se construyen de manera que estos movimientos sean muy pequeños, típicamente entre 1nm y 1µm por cada voltio aplicado, dependiendo del material y su geometría. Para hacernos una idea del orden de magnitud, la distancia entre átomos, aunque depende del material, suele ser del orden de algunas décimas de nm, por lo que muchos de estos actuadores son capaces de realizar movimientos subatómicos y, si la interacción lo permite, un SPM puede “ver” átomos.

### 6.7.2 Modo *tapping*®, contacto intermitente o modo dinámico

El AFM emplea como sonda una micropalanca terminada en una punta afilada, que es la que palpa la superficie como haría una persona invidente con su bastón. Este modo de funcionamiento se suele denominar modo de contacto (*contact mode*) y presenta el inconveniente de aplicar fuerzas relativamente altas que pueden dañar la superficie si es muy delicada o desplazar los objetos depositados sobre ella si no están bien adheridos. Volviendo a la analogía del bastón, podemos pensar en qué sucede si se intenta palpar una pelota depositada sobre el suelo. Si el suelo es plano y la masa de la pelota no es excesiva, es muy probable que al pasar el bastón por ella la desplacemos, algo indeseable si lo que queremos analizar es la propia pelota. A lo largo de las últimas décadas se han desarrollado diversas técnicas para reducir estas fuerzas, entre ellas el modo de contacto intermitente (*tapping mode*) también llamado modo dinámico.

En el modo de contacto intermitente la micropalanca oscila en su frecuencia de resonancia, inicialmente cerca de la muestra, pero sin llegar a tocarla. Si esta distancia se reduce progresivamente, llegará un momento en el que impacte en el punto más cercano de la oscilación, sin llegar a estar ambos elementos continuamente en contacto, de ahí el nombre de “contacto intermitente”. La frecuencia de la oscilación suele variar entre las decenas y centenas de kHz, llegando en algunos casos a los MHz, y las amplitudes varían entre pocos nm y centenas de ellos. Cuando se produce el impacto

con la muestra, la palanca pierde energía y la amplitud de la oscilación se reduce de manera aproximadamente lineal según se acerca la punta a la muestra. Si empleamos el amplificador lock-in para detectar esta amplitud podemos controlar la distancia, alejando los elementos si la amplitud medida es menor que la deseada y acerándolos si es mayor. La Figura 56 muestra medidas reales de la dependencia de la amplitud con la posición Z promedio de la punta. El eje de ordenadas aumenta según se reduce la distancia punta-muestra y tiene su origen en el punto elegido como consigna para realizar la adquisición de las imágenes. Cuando la distancia es mayor (valores negativos) la amplitud a la que oscila la palanca también lo es, llegando a una zona teóricamente plana, donde no hay interacción. En esta zona el sistema es más sensible al ruido, debido a que la oscilación tiene menos amortiguación. A la derecha, donde la dependencia de la amplitud con la distancia es prácticamente lineal, es donde se puede controlar la distancia manteniendo constante la amplitud mediante el acercamiento o alejamiento de la punta a la muestra. Para minimizar la fuerza aplicada interesa que la consigna se elija cerca de la amplitud de oscilación libre y para disponer de un mayor rango de respuesta que sea lejos de este valor, por lo que hay que buscar un valor de compromiso.

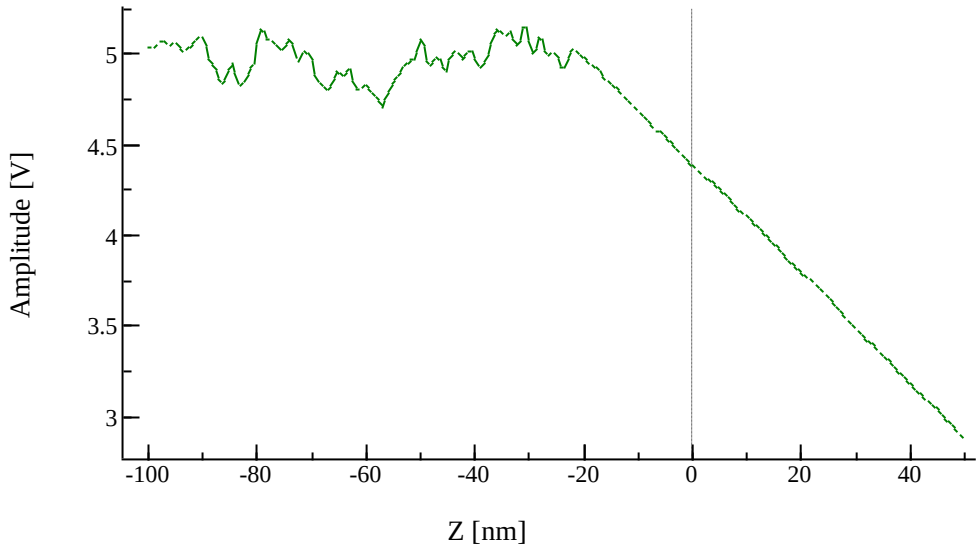



**Figura 56. Medida de la amplitud de oscilación en función de la distancia**

### 6.7.3 Medidas realizadas con el AFM

Hemos adquirido dos conjuntos de imágenes con distintas micropalancas, obteniendo un resultado satisfactorio en ambos casos. Primero hemos medido una superficie que se emplea para calibrar los actuadores piezoeléctricos, consistente en un patrón de litografía con un periodo conocido, en este caso 3µm. Al ser este tamaño conocido se puede calcular el factor de conversión de V a nm del actuador. Estas medidas se han realizado con una micropalanca de aproximadamente 70kHz de frecuencia de resonancia. Se puede ver el resultado de la medida en la Figura 57, que muestra la imagen de alturas sin ningún tipo de procesado posterior junto a la paleta usada para el eje Z. El valor de altura cero es arbitrario y es común ajustarlo al mínimo de los datos de la imagen.

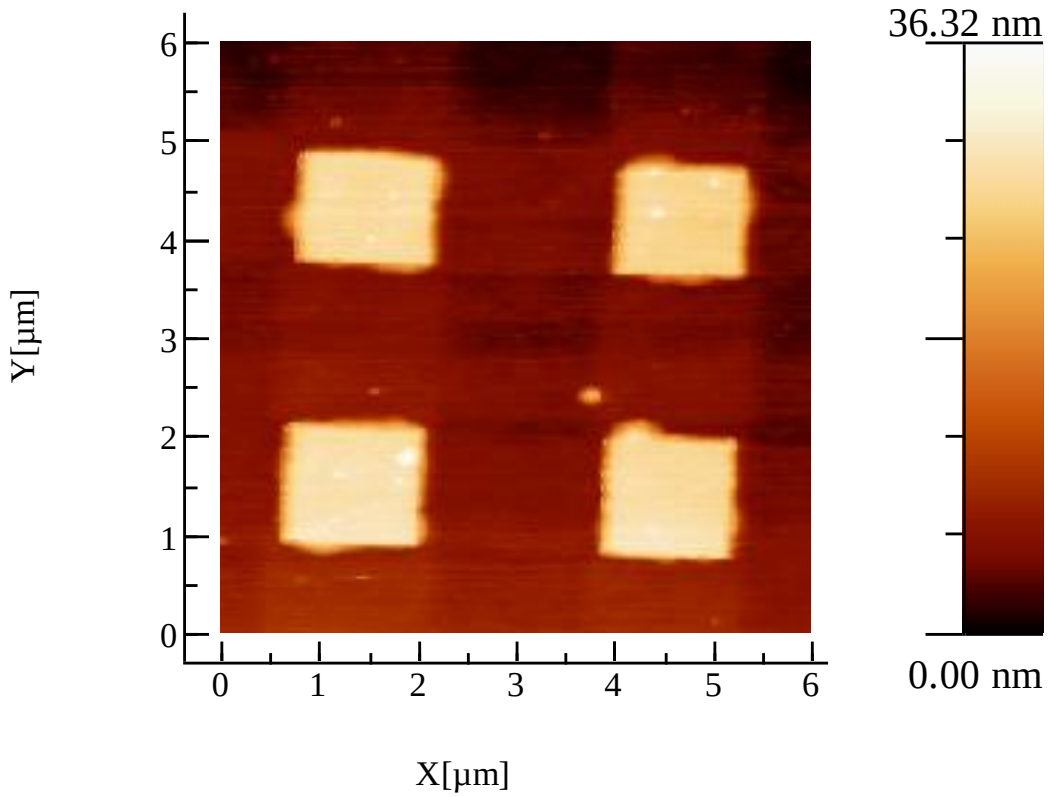



**Figura 57. Medida de una muestra de calibración**

La segunda prueba, quizá más espectacular para los profanos en el campo y en cualquier caso más exigente para el amplificador lock-in por trabajar a mayor frecuencia y menor amplitud de la señal a analizar, se realizó midiendo una muestra muy plana con una micropalanca de unos 240kHz de frecuencia de resonancia y pocas decenas de mV de amplitud de la señal a analizar, lo que nos obligaba a emplear los amplificadores de tensión de la entrada, concretamente con ganancia 100V/V. La muestra medida es grafito pirolítico altamente orientado, consistente en capas de átomos de carbono depositadas unas sobre otras como si fueran sábanas de grafeno de 0.34nm de grosor. Estas medidas son más complicadas de realizar, no solo por las exigencias al amplificador lock-in, sino porque en estos valores de distancias cualquier vibración, incluso acústica, puede producir interferencias de una magnitud comparable a las alturas que se quieren medir. En general los AFMs no son capaces de resolver átomos individuales, o si lo son, necesitan unas condiciones ambientales muy exigentes, como la ausencia casi absoluta de aire, requiriendo estar dentro de campanas de ultra alto vacío. Los datos que presentamos se midieron a temperatura ambiente y presión atmosférica. Se puede ver una de las imágenes en la Figura 58. Por su altura total deducimos que tiene unas 20 capas de átomos carbono de diferencia entre las partes más altas y las más bajas.

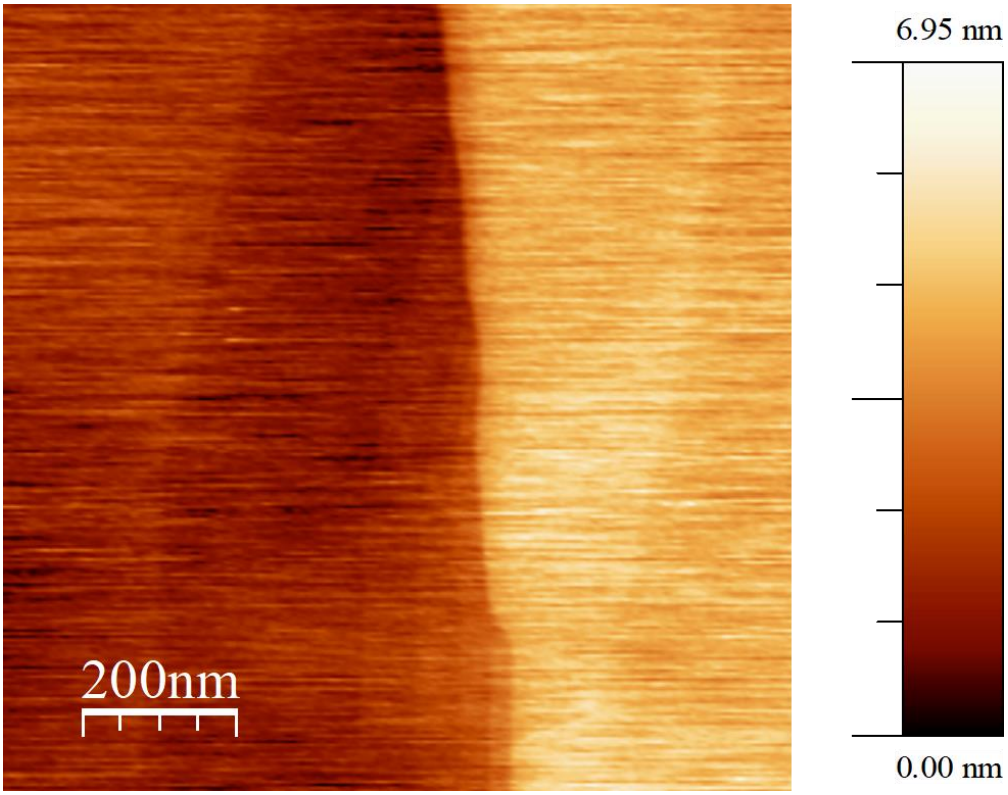



**Figura 58. Imagen de grafito pirolítico altamente orientado**

En la zona inferior central se observan pequeños escalones, cuyo borde a veces se confunde con el ruido de la medida y que por el poco contraste que presentan unos con otros bien podrían ser de un átomo de altura. Si realizamos un perfil de alturas filtrando el ruido en esta zona mediante el promediado de distintas líneas obtenemos la curva de la Figura 59. En ella podemos apreciar tres escalones cuya altura coincide bastante bien con la esperada para escalones monoatómicos, por lo que podemos concluir que efectivamente lo son. Estas medidas se repitieron empleando un sistema comercial y los resultados que se obtuvieron fueron similares a los que se presentan aquí.

Para la comunicación del amplificador lock-in con el controlador del AFM se han empleado las salidas analógicas de resultados, mucho más rápidas, aunque con pérdida de calidad. Los resultados indican que la pérdida de calidad en esta configuración no es preocupante.

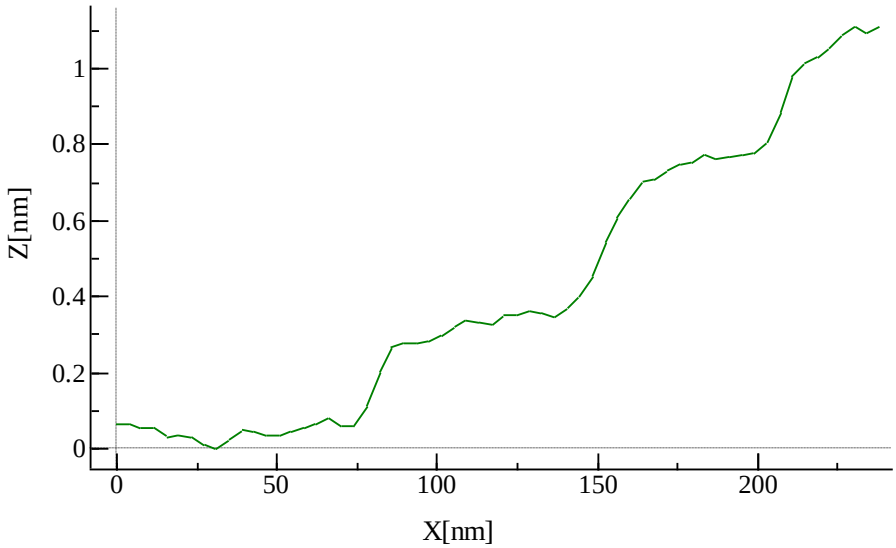



**Figura 59. Perfil de alturas sobre el grafito**

## 6.8 Imágenes del prototipo terminado

A continuación se muestran dos fotos del prototipo terminado, una vista superior sin tapa ni la carátula trasera (Figura 60), donde se aprecia la distribución interna de los componentes, y una vista frontal con todos los elementos (Figura 61).


**Figura 60. Versión final del dispositivo desarrollado (vista superior sin cubierta ni tapa trasera)**


**Figura 61. Vista frontal del dispositivo desarrollado**

# CAPÍTULO 7. CONCLUSIONES

En este capítulo vamos a realizar un breve análisis sobre los resultados obtenidos y las posibilidades de trabajo futuro que abre este TFM.

## 7.1 Resumen

1. En este trabajo hemos diseñado, simulado, construido y verificado un amplificador lock-in que usa un microcontrolador como procesador digital de señales y unas etapas analógicas sencillas que permiten configurar ciertos parámetros para mejorar los resultados.
2. El trabajo se puede considerar multidisciplinar en sí mismo, al igual que lo es el máster universitario del que forma parte. Para realizarlo ha sido necesario poner en práctica principalmente conocimientos de sistemas empotrados, procesado de señales y modelado y simulación de sistemas, pero también de caracterización de sistemas, electrónica analógica y digital, sistemas híbridos y otros conceptos estudiados durante el máster o en etapas previas.
3. Se han aplicado técnicas de gestión de proyectos, definiendo unas especificaciones previas y realizando el resto de las tareas en base a ellas.
4. Para la elección de componentes nos enfrentamos a un problema típico de ingeniería: buscar el mejor compromiso entre especificaciones, precio y tiempo de desarrollo en base al objetivo que se quiere cumplir.
5. El dispositivo se ha empleado como parte de un sistema de control más complejo, obteniendo resultados comparables a la versión comercial.
6. El resultado final ha sido un dispositivo que no solo demuestra que el concepto de partida es perfectamente válido, sino que puede competir con algunos aparatos comerciales en cuanto a especificaciones técnicas y resultados de las pruebas.

## 7.2 Mejoras y trabajo futuro

Aunque el Trabajo Fin de Máster constituye el final de una etapa, como su propio nombre indica, consideramos que el amplificador desarrollado aún tiene margen de mejora, aunque sea en un contexto distinto. Las propuestas de trabajo futuro identificadas son:

1. Las correcciones y mejoras al diseño *hardware* identificadas y listadas en el apartado 6.1.2.
2. Reemplazar los amplificadores operacionales por otros de mejores características dinámicas.
3. Realizar una caracterización completa del ruido y revisar, y si fuera necesario corregir, las etapas analógicas.
4. Añadir el control y transmisión de resultados por ethernet, aprovechando que el microcontrolador implementa esta interfaz, para obtener más ancho de banda.
5. Añadir funcionalidad adicional aprovechando el desarrollo *hardware*, tal como adquirir bloques de datos a alta frecuencia para un procesado posterior (FFT o similar), opción de varios demoduladores, etc.

6. Añadir la opción de sumar una señal DC a la salida de excitación para ampliar los usos posibles.
7. Prescindir del módulo de evaluación del microcontrolador y realizar un diseño de placa que en su lugar incluya el circuito integrado y sus componentes auxiliares.
8. Mejorar la eficiencia de partes críticas empleando código ensamblador, por ejemplo, en el bucle del núcleo M4.
9. Revisar los módulos que implementa por *hardware* el microcontrolador para intentar mejorar la eficiencia.

## BIBLIOGRAFÍA


[1] Arduino S.r.l., “Arduino - Home.” https://www.arduino.cc/ (accessed Feb. 21, 2022).

[2] The Raspberry Pi Foundation, “Teach, Learn, and Make with Raspberry Pi.” https://www.raspberrypi.org/ (accessed Feb. 21, 2022).

[3] GitHub Inc, “GitHub: Where the world builds software · GitHub.” https://github.com/ (accessed Oct. 11, 2020).

[4] Digi-Key Electronics, “DigiKey Electronics España.” https://www.digikey.es/ (accessed Feb. 21, 2022).

[5] Mouser Electronics Inc, “Distribuidor de Componentes Electrónicos - Mouser Electronics España.” https://www.mouser.es/ (accessed Feb. 21, 2022).

[6] Amidata S.A.U., “RS Components | Su partner digital de soluciones industriales.” https://es.rs-online.com/web/ (accessed Feb. 21, 2022).

[7] R. Dicke, “Oral History Interviews | Robert Dicke | American Institute of Physics.” https://www.aip.org/history-programs/niels-bohr-library/oral-histories/4572 (accessed Mar. 16, 2022).

[8] C. R. Cosens, “A balance-detector for alternating-current bridges,” *Proceedings of the Physical Society*, vol. 46, no. 6, pp. 818–823, Nov. 1934, doi: 10.1088/0959-5309/46/6/310.

[9] W. C. Michels and N. L. Curtis, “A Pentode Lock-In Amplifier of High Frequency Selectivity,” *Review of Scientific Instruments*, vol. 12, no. 9, pp. 444–447, Sep. 1941, doi: 10.1063/1.1769919.

[10] R. Fernandez Molanes, K. Amarasinghe, J. Rodriguez-Andina, and M. Manic, “Deep learning and reconfigurable platforms in the internet of things: Challenges and opportunities in algorithms and hardware,” *IEEE Industrial Electronics Magazine*, vol. 12, no. 2, pp. 36–49, Jun. 2018, doi: 10.1109/MIE.2018.2824843.

[11] A. A. Dorrington and R. Künnemeyer, “A simple microcontroller based digital lock-in amplifier for the detection of low level optical signals,” *Proceedings - 1st IEEE International Workshop on Electronic Design, Test and Applications, DELTA 2002*, pp. 486–488, 2002, doi: 10.1109/DELTA.2002.994680.

[12] Zurich Instruments AG, “UHFLI 600 MHz Lock-in Amplifier | Zurich Instruments.” https://www.zhinst.com/others/en/products/uhfli-lock-in-amplifier (accessed Jul. 13, 2022).

[13] Liquid Instruments, “Lock-in Amplifier (Comparison) - Liquid Instruments.” https://www.liquidinstruments.com/products/integrated-instruments/lock-in-amplifier-comparison/ (accessed Jul. 13, 2022).

[14] S. C. Hageman and / Analoghome, “A Modern DSP Based Lock-In Amplifier Designed for Code and Hardware Experimentation,” 2018.

[15] Liquid Instruments, “Lock-in Amplifier (Moku:Pro) - Liquid Instruments.” https://www.liquidinstruments.com/products/integrated-instruments/lock-in-amplifier-mokupro/ (accessed Mar. 16, 2022).

[16] Y. Cao, B. Sun, and D. He, “Measurement and Research of Beam-Based Alignment With AC k-Modulation at HLS,” *IEEE Trans Nucl Sci*, vol. 54, no. 2, pp. 367–374, Apr. 2007, doi: 10.1109/TNS.2007.891620.

[17] AMETEK Scientific Instruments, “7270 General Purpose DSP Lock-in Amplifier | Signal Recovery.” https://www.ameteksi.com/support-center/legacy-products/signal-recovery-legacy/lock-in-amplifier-legacy/7270-general-purpose-dsp-lock-in-amplifier (accessed Jul. 16, 2022).
[18] H. Elmqvist, S. E. Mattsson, and M. Otter, “Modelica - an international effort to design an object-oriented modelling language,” *{Proceedings of the 1998 Summer Computer Simulation Conference}*, pp. 333–339, 1998.
[19] H. E. S. M. Karl Johan Åström, “Evolution of Continuous-Time Modeling and Simulation.,” *Esm*, pp. 1–10, 1998, [Online]. Available: http://www.modelica.org/publications/index_html/papers/esm98his.pdf
[20] The Modelica Association, “Modelica Libraries — Modelica Association.” https://modelica.org/libraries.html (accessed Jul. 20, 2022).
[21] P. Fritzson *et al.*, “The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development,” *Identification and Control*, vol. 41, no. 4, pp. 241–285, 2020, doi: 10.4173/mic.2020.4.1.
[22] D. Prentice, “Arduino Due digitalWrite vs. direct port manipulation speed - Hardware / Arduino Due - Arduino Forum.” https://forum.arduino.cc/t/arduino-due-digitalwrite-vs-direct-port-manipulation-speed/666404 (accessed Jul. 19, 2022).
[23] ST Microelectronics, “STMicroelectronics.” https://www.st.com/ (accessed Jul. 19, 2022).
[24] ST Microelectronics, “STM32H7 - Arm Cortex-M7 and Cortex-M4 MCUs (480 MHz) - STMicroelectronics.” https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-high-performance-mcus/stm32h7-series.html (accessed Jul. 19, 2022).
[25] Altium Limited, “Software y herramientas de diseño de PCB para construir la próxima generación en electrónica | Altium.” https://www.altium.com/es (accessed Feb. 21, 2022).
[26] G. Binnig, C. F. Quate, and Ch. Gerber, “Atomic Force Microscope,” *Phys Rev Lett*, vol. 56, no. 9, pp. 930–933, Mar. 1986, doi: 10.1103/PhysRevLett.56.930.
[27] “Nanoforces | Nanoforces.” https://www.nanoforces.es/ (accessed Aug. 20, 2022).
[28] I. Horcas, R. Fernández, J. M. Gómez-Rodríguez, J. Colchero, J. Gómez-Herrero, and A. M. Baro, “WSXM: A software for scanning probe microscopy and a tool for nanotechnology,” *Review of Scientific Instruments*, vol. 78, no. 1, 2007, doi: 10.1063/1.2432410.
[29] G. Binnig and H. Rohrer, “The Scanning Tunneling Microscope,” *Sci Am*, vol. 253, no. 2, pp. 50–56, Aug. 1985, doi: 10.1038/scientificamerican0885-50.
[30] Nobel Prize Outreach AB, “NobelPrize.org.” https://www.nobelprize.org/prizes/lists/all-nobel-prizes/ (accessed Aug. 21, 2022).

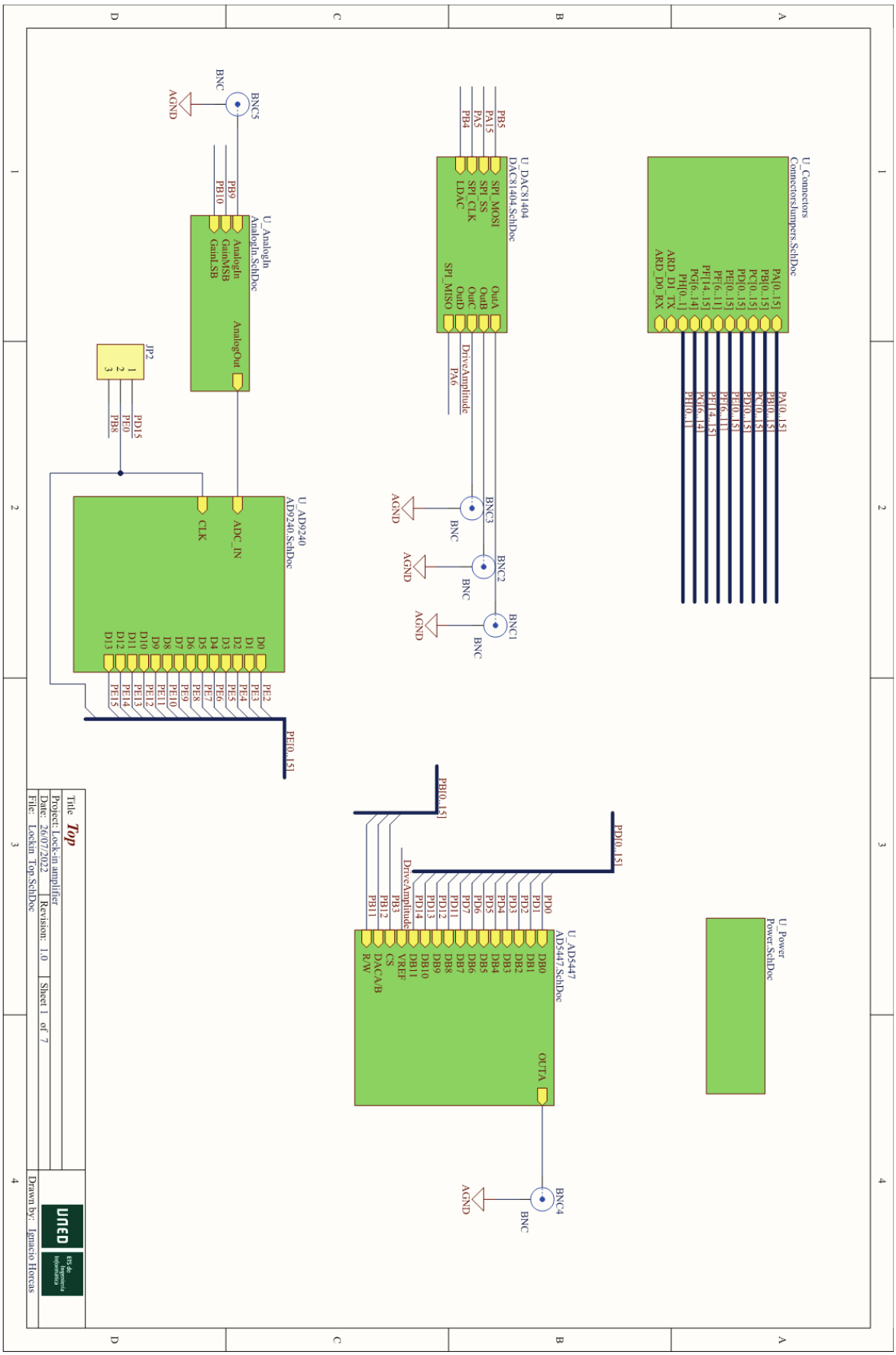
## LISTA DE SÍMBOLOS

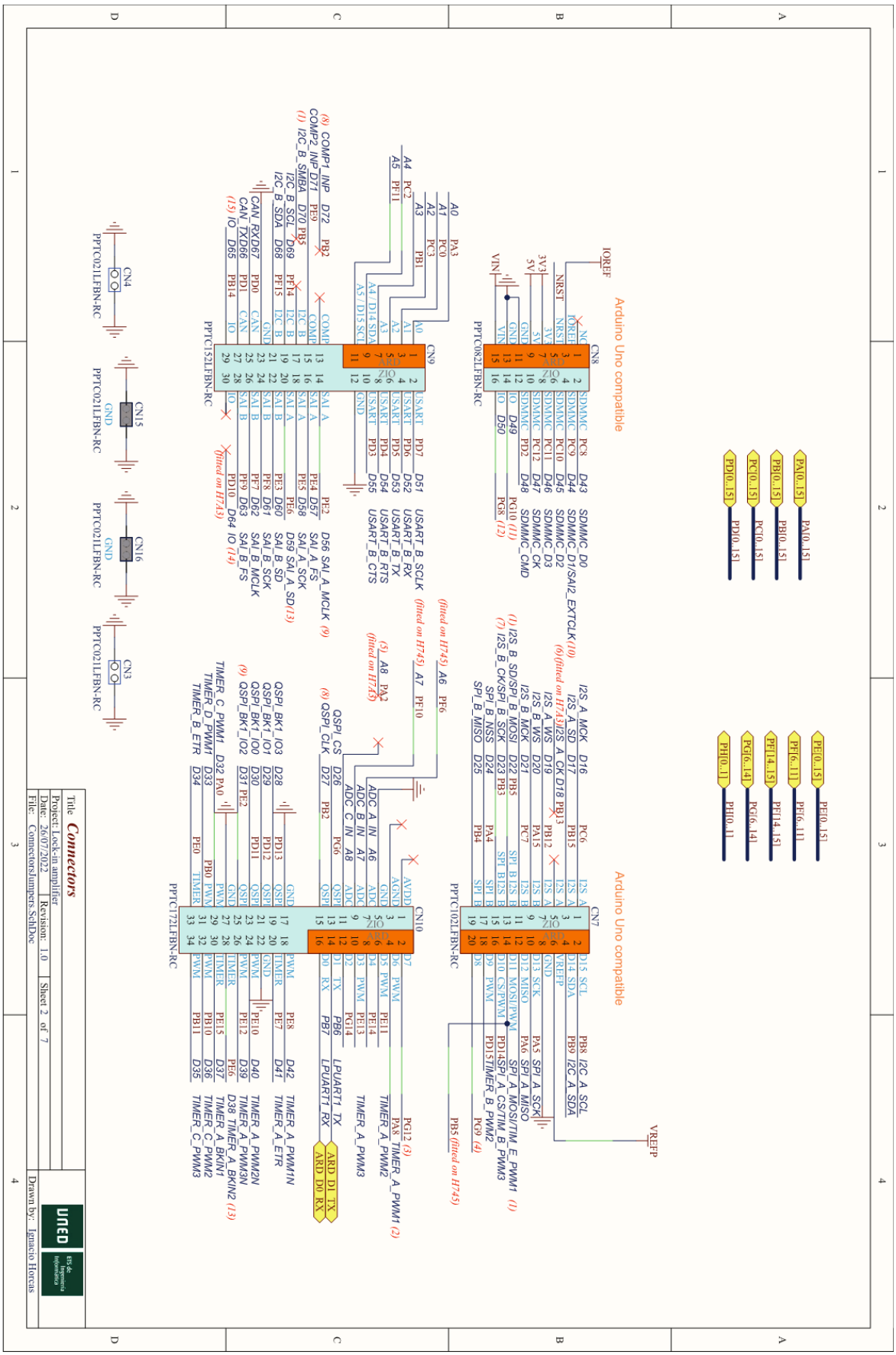
| | |
|---|---|
| ADC | Analog to Digital Converter (Convertidor de analógico a digital) |
| AFM | Atomic Force Microscope (Microscopio de fuerzas atómicas) |
| API | Application Programming Interface (Interfaz de programación de aplicación) |
| ASCII | American Standard Code for Information Interchange (Código estándar estadounidense para el intercambio de información) |
| BNC | Bayonet Neil-Concelman (Bayoneta Neil-Concelman) |
| BSD | Berkeley Software Distribution (Distribución de *software* Berkeley) |
| CC | Corriente Continua |
| CI | Circuito Integrado |
| CPU | Central Proccess Unit (Unidad de procesado central) |
| CS | Chip Select (selección de componente) |
| DIN | Deutsche Industrial Norms (Normas industriales alemanas) |
| DIY | Do It Yourself (Bricolaje) |
| DMA | Direct Memory Access (Acceso directo a memoria) |
| ECAD | Electronics Computer Aid Design (Diseño electrónico asistido por ordenador) |
| EDA | Ecuación Diferencial Algebraica |
| EDO | Ecuación Diferencial Ordinaria |
| FFT | Fast Fourier Transform (Transformada Rápida de Fourier) |
| FIR | Finite Impulse Response (Respuesta finita a un impulso) |
| FPGA | Field Programmable Gate Array (Matriz de puertas programable en campo) |
| GPIO | General Purpose Input-Output (Entrada-salida de propósito general) |
| $I^2C$ | Inter-Integrated Circuit (Circuito inter-integrado) |
| IDE | Integrated Development Environment (Entorno de desarrollo integrado) |
| IIR | Infinite Impulse Response (Respuesta infinita a un impulso) |
| IoT | Internet of Things (Internet de las cosas) |
| GNU | GNU is Not Unix (Acrónimo recursivo: GNU no es Unix) |
| LED | Light Emitting Diode (Diodo emisor de luz) |
| LVTTL | Low Voltage Transistor-Transistor Logic (Lógica transistor-transistor de bajo voltaje) |
| MSL | Modelica Standar Library (Biblioteca estándar de Modelica) |
| TFM | Trabajo Fin de Máster |
| TTL | Transistor-Transistor Logic (Lógica transistor-transistor) |
| PCB | Printed Circuit Board (Placa de circuito impreso) |
| RC | Resistencia-Condensador |
| RPC | Remote Procedure Call (Llamada a procedimiento remoto) |
| SCPI | Standard Commands for Programmable Instruments (Comandos normalizados para instrumentos programables) |
| SMA | SubMiniature version A (Subminiatura versión A) |
| SNR | Signal Noise Ratio (relación señal/ruido) |
| SoC | Sistem on Chip (Sistema contenido en un único circuito integrado) |
| SPI | Serial Peripheral Interface (Interfaz serie periférica) |
| SPM | Scanning Probe Microscope (Microscopio de sonda de barrido) |
| STM | Scanning Tunnel Microscope (Microscopio de efecto túnel) |

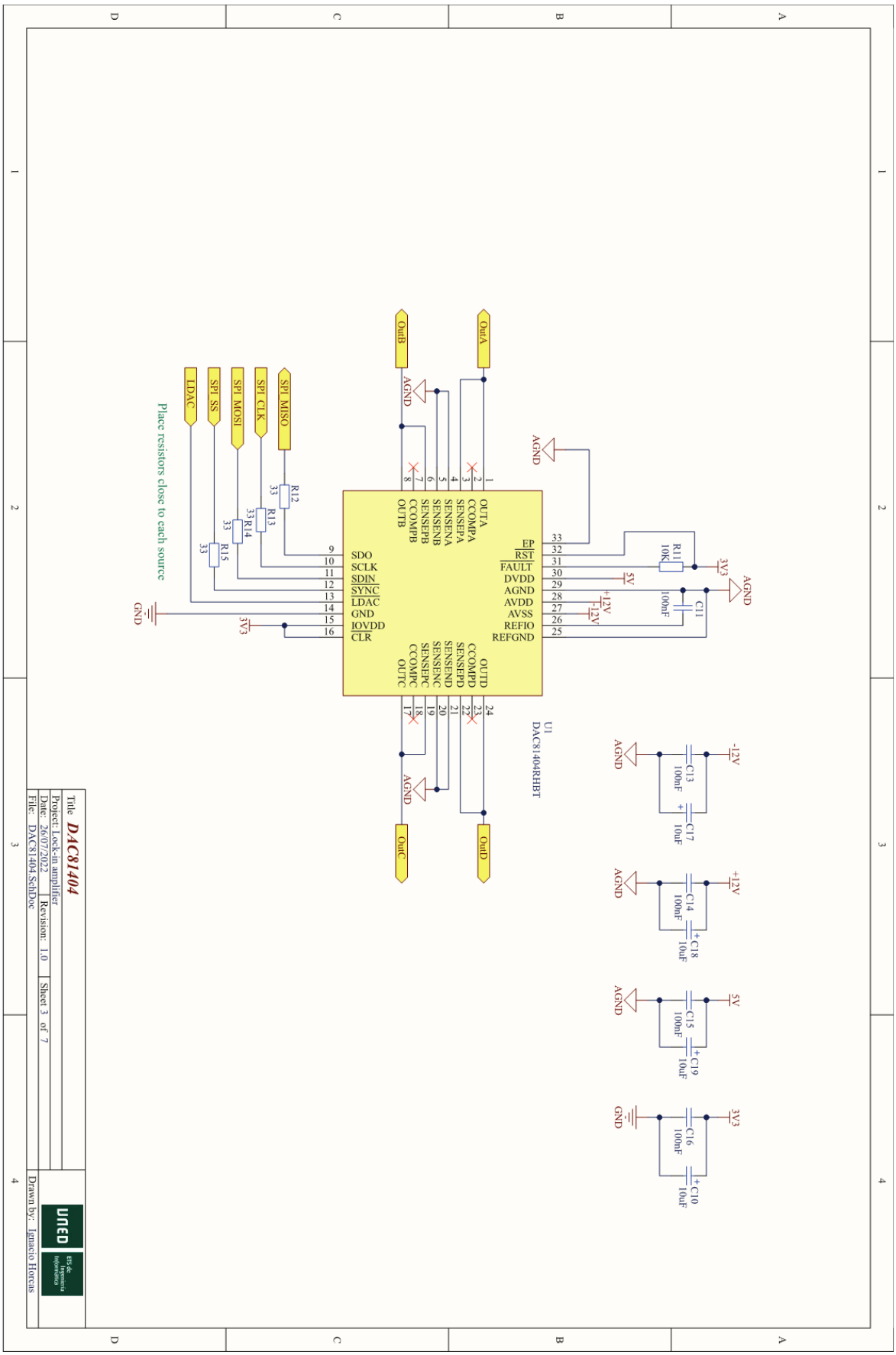
| | |
|---|---|
| UART | Universal Asynchronous Receiver-Transmitter (Transmisor-receptor asíncrono universal) |
| UCM | Universidad Complutense de Madrid |
| UNED | Universidad Nacional de Educación a Distancia |
| USB | Universal Serial Bus (Bus serie universal) |
| WDT | Watch Dog Timer (Temporizador de perro guardián) |

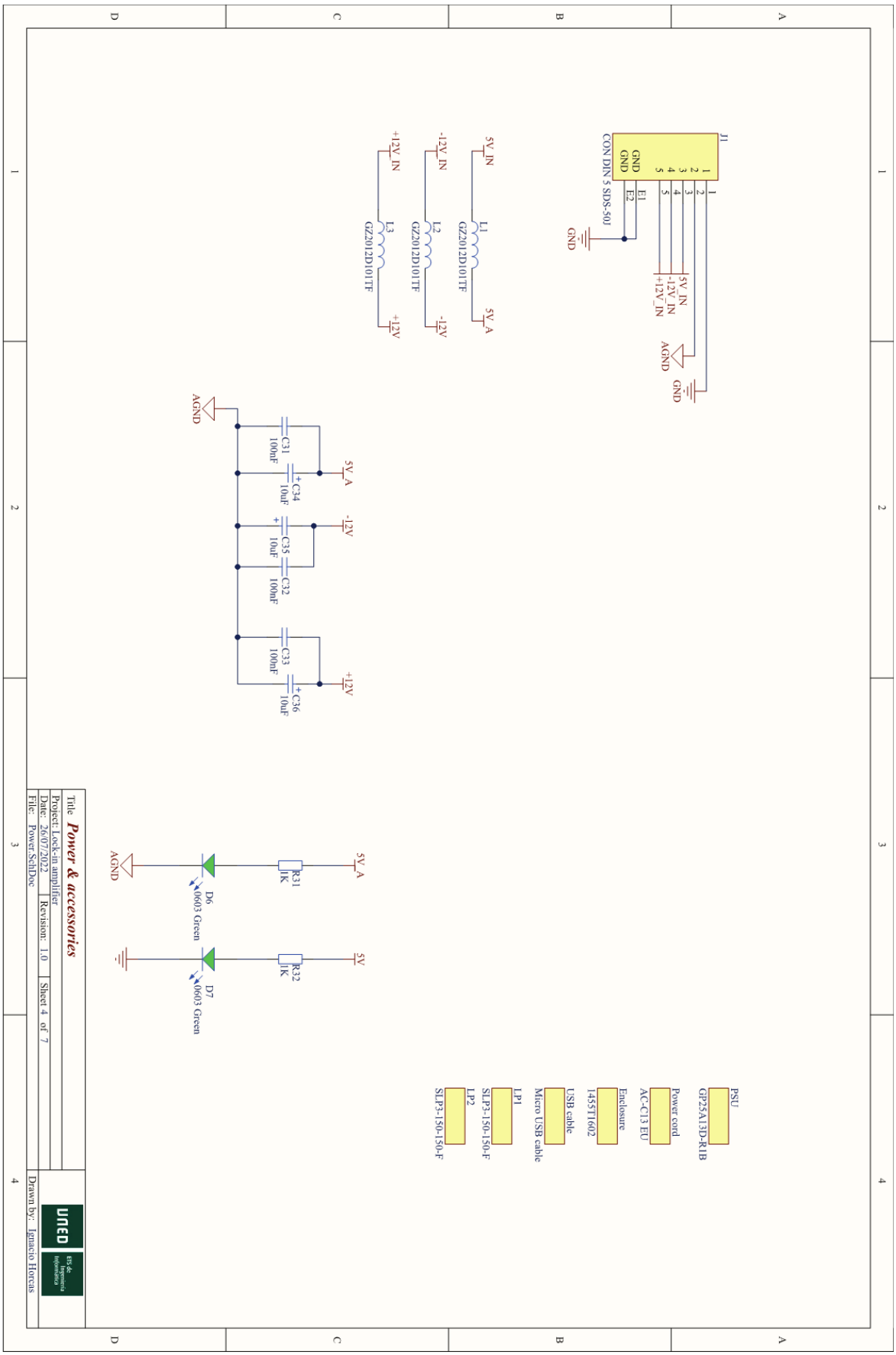
## APÉNDICE I. Esquemáticos

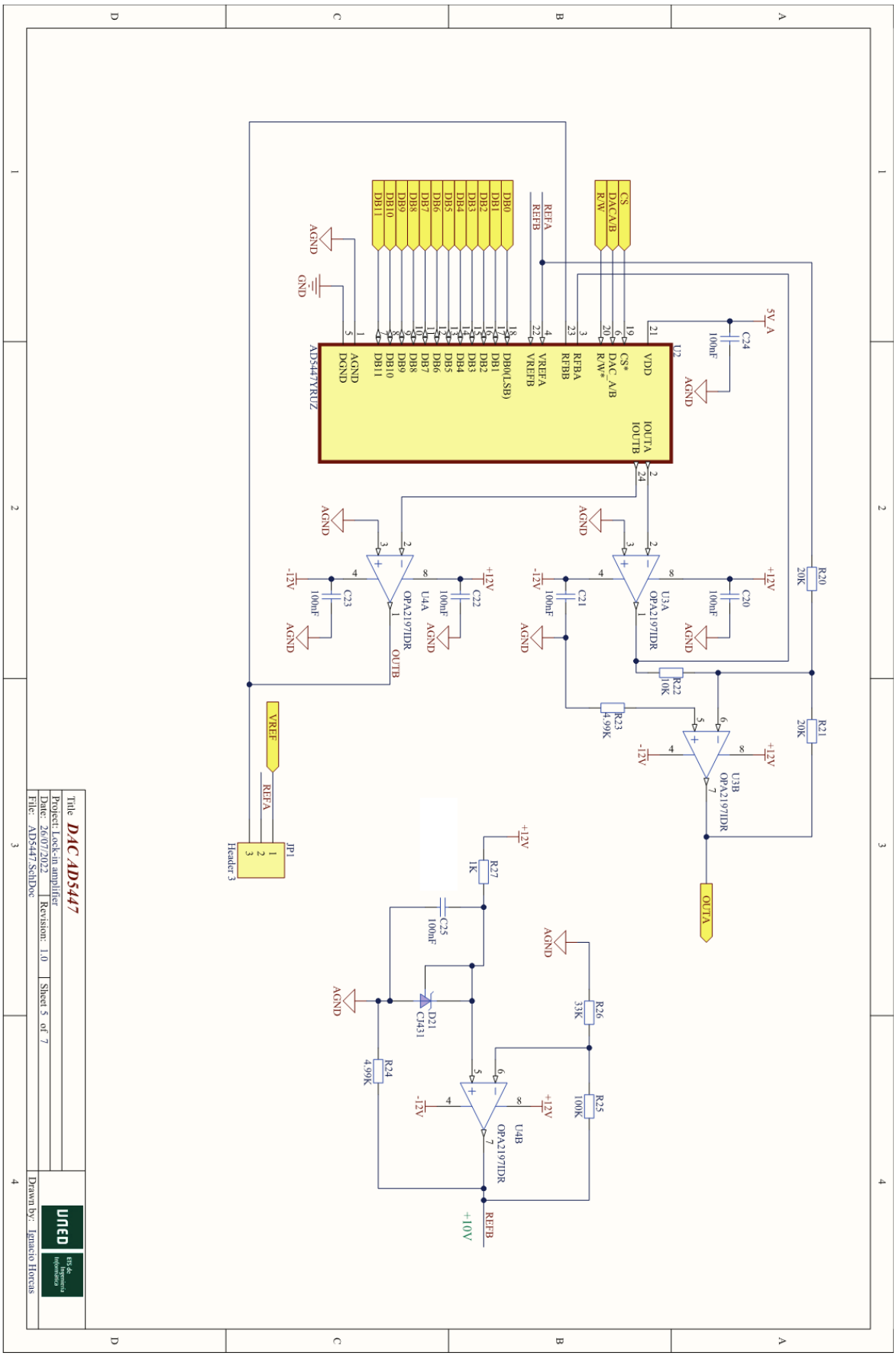
En este apéndice se incluyen los esquemáticos de la placa de circuito impreso que han dado lugar al prototipo desarrollado. No incluyen las correcciones ni mejoras detectadas durante la etapa de validación.

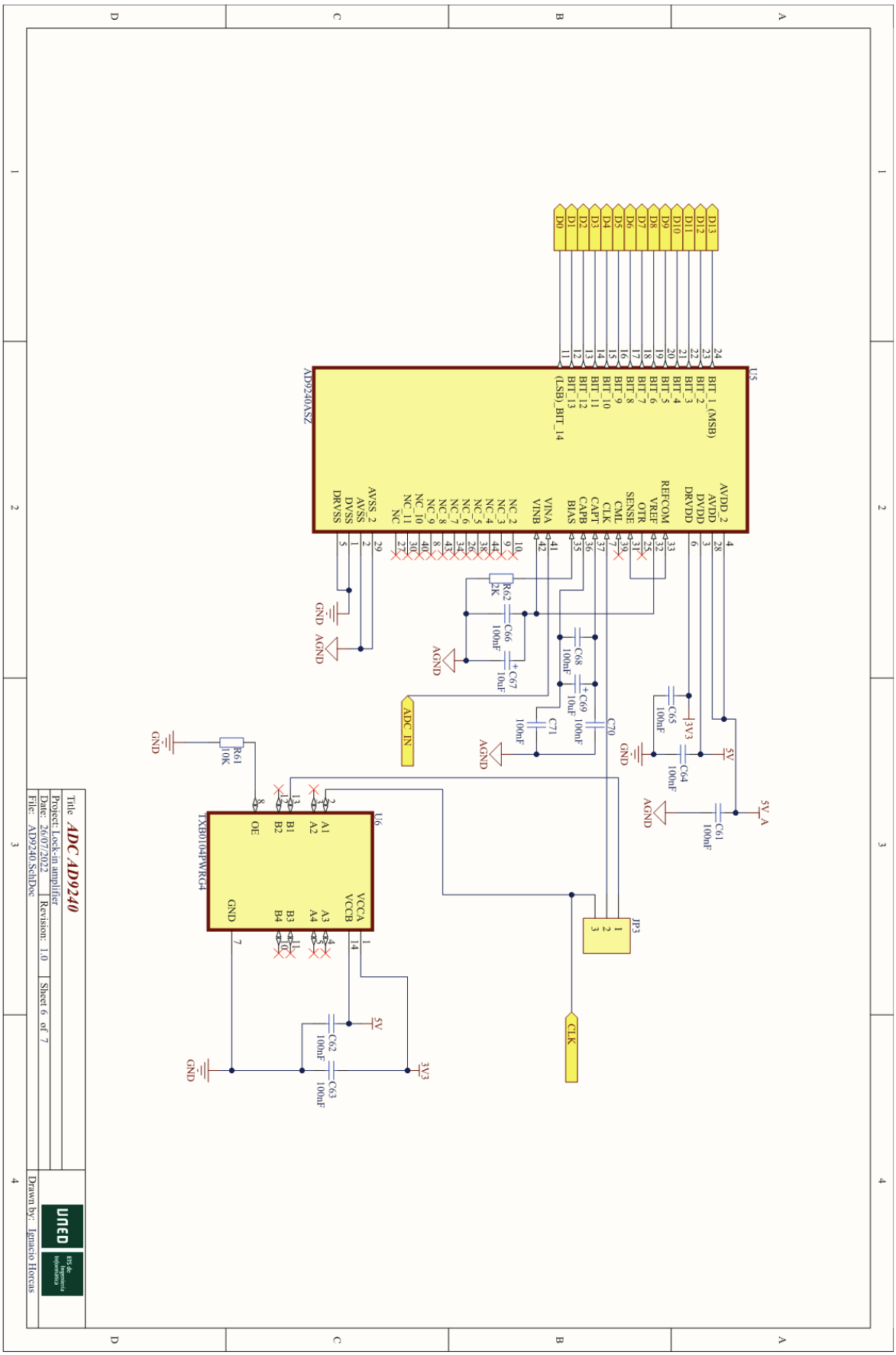

U_Connectors
ConnectorsJumpers.SchDoc
PA[0..15]
PB[0..15]
PC[0..15]
PD[0..15]
PE[0..15]
PF[6..11]
PF[14..15]
PG[6..14]
PH[0..1]
ARD_D1_TX
ARD_D0_RX
U_DAC81404
DAC81404.SchDoc
SPI_MOSI
SPI_SS
SPI_CLK
LDAC
SPI_MISO
OutA
OutB
OutC
OutD
DriveAmplitude
PA6
PB5
PA15
PA5
PB4
U_AnalogIn
AnalogIn.SchDoc
AnalogIn
GainMSB
GainLSB
AnalogOut
PB9
PB10
BNC5
BNC
AGND
JP2
PD15
PE0
PB8
U_AD9240
AD9240.SchDoc
CLK
ADC_IN
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
PE2
PE3
PE4
PE5
PE6
PE7
PE8
PE9
PE10
PE11
PE12
PE13
PE14
PE15
PE[0..15]
BNC3
BNC2
BNC1
PB[0..15]
PD[0..15]
U_AD5447
AD5447.SchDoc
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
DB8
DB9
DB10
DB11
VREF
CS
DACA/B
R/W
OUTA
BNC4
U_Power
Power.SchDoc
Title
Top
Project: Lock-in amplifier
Date: 26/07/2022
Revision: 1.0
Sheet 1 of 7
File: Lockin Top.SchDoc
Drawn by: Ignacio Horcas
ETS de Ingeniería Informática

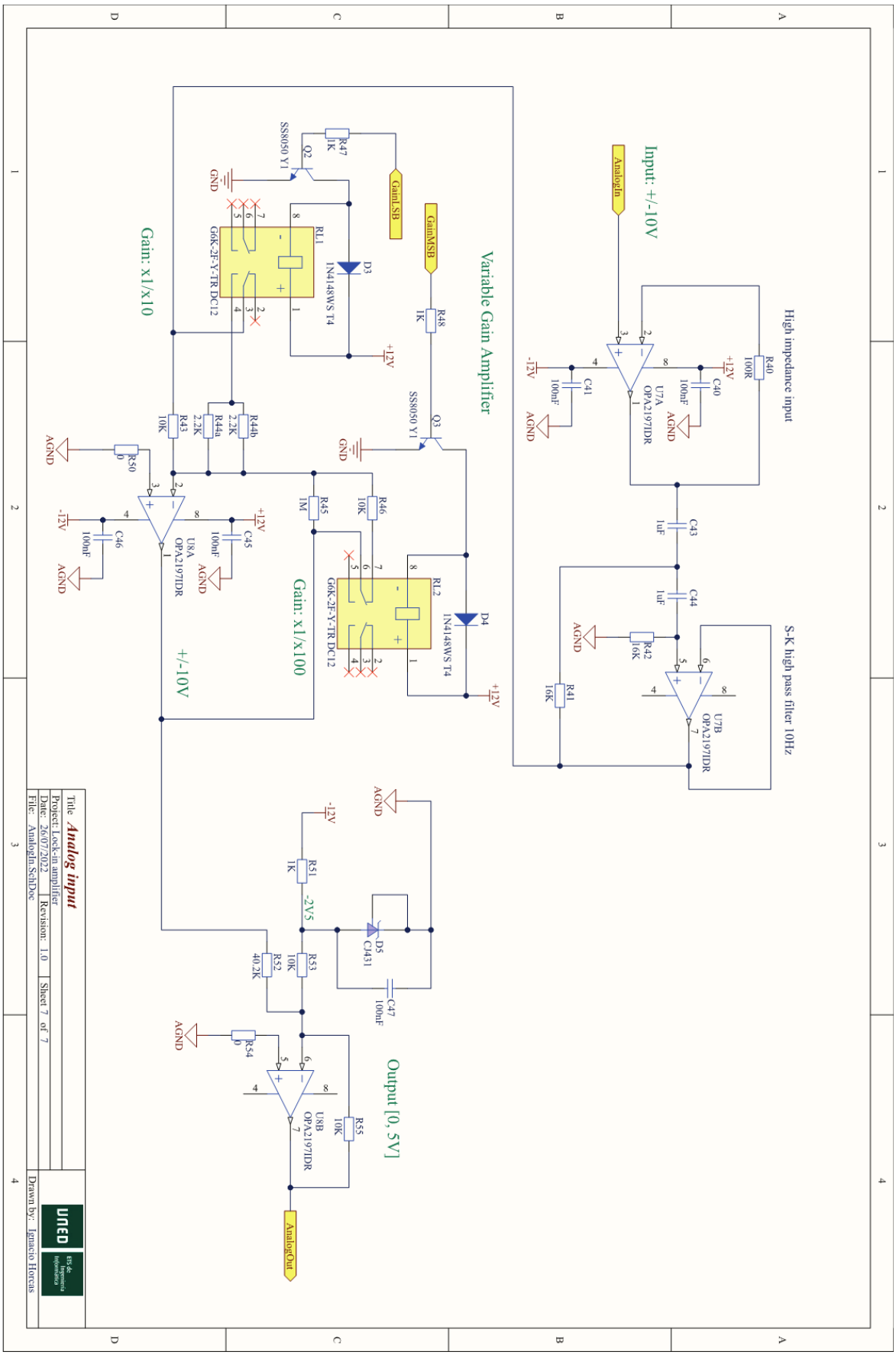

Arduino Uno compatible
Arduino Uno compatible
Connectors
Project: Lock-in amplifier
Date: 26/07/2022
Revision: 1.0
Sheet 2 of 7
File: Connectors\Jumpers.SchDoc
Drawn by: Ignacio Horcas
ETS de Ingeniería Informática


Place resistors close to each source
OutA
OutB
OutC
OutD
SPI MISO
SPI CLK
SPI MOSI
SPI SS
LDAC
R12
R13
R14
R15
U1
DAC81404RHBT
SDO
SCLK
SDIN
SYNC
LDAC
GND
IOVDD
CLR
EP
RST
FAULT
DVDD
AGND
AVDD
AVSS
REFIO
REFGND
R11
10K
C11
100nF
C13
C14
C15
C16
C17
C18
C19
C10
10uF
3V3
5V
+12V
-12V
GND
Title DAC81404
Project: Lock-in amplifier
Date: 26/07/2022
Revision: 1.0
Sheet 3 of 7
File: DAC81404.SchDoc
Drawn by: Ignacio Horcas


J1
CON DIN 5 SDS-50J
GND
GND
E1
E2
5V_IN
-12V_IN
+12V_IN
AGND
GND
5V_IN
L1
GZ2012D101TF
5V_A
-12V_IN
L2
GZ2012D101TF
-12V
+12V_IN
L3
GZ2012D101TF
+12V
AGND
C31
100nF
C34
10uF
5V_A
C35
10uF
C32
100nF
-12V
C33
100nF
C36
10uF
+12V
AGND
D6
0603 Green
R31
1K
5V_A
D7
0603 Green
R32
1K
5V
PSU
GP25A13D-R1B
Power cord
AC-C13 EU
Enclosure
1455T1602
USB cable
Micro USB cable
LP1
SLP3-150-150-F
LP2
SLP3-150-150-F
Title Power & accessories
Project: Lock-in amplifier
Date: 26/07/2022
Revision: 1.0
Sheet 4 of 7
File: Power.SchDoc
Drawn by: Ignacio Horcas
UNED
ETS de Ingeniería Informática


DAC AD5447
Project: Lock-in amplifier
Date: 26/07/2022
Revision: 1.0
Sheet 5 of 7
File: AD5447.SchDoc
Drawn by: Ignacio Horcas
AD5447YRUZ
OPA2197IDR
REFB
+10V
OUTA
VREF
REFA
Header 3


ADC AD9240
Project: Lock-in amplifier
Date: 26/07/2022
Revision: 1.0
Sheet 6 of 7
File: AD9240.SchDoc
Drawn by: Ignacio Horcas
AD9240ASZ
TXB0104PWRG4
ADC_IN
CLK
5V_A
3V3
5V
GND
AGND


Input: +/-10V
AnalogIn
High impedance input
R40
100R
U7A
OPA2197IDR
C40
100nF
C41
100nF
+12V
-12V
AGND
C43
1uF
C44
1uF
R42
16K
R41
16K
S-K high pass filter 10Hz
U7B
OPA2197IDR
Variable Gain Amplifier
GainLSB
GainMSB
R48
1K
Q3
SS8050 Y1
Q2
SS8050 Y1
R47
1K
GND
RL1
G6K-2F-Y-TR DC12
D3
1N4148WS T4
RL2
G6K-2F-Y-TR DC12
D4
1N4148WS T4
Gain: x1/x10
Gain: x1/x100
R43
10K
R44a
2.2K
R44b
2.2K
R45
1M
R46
10K
R50
0
U8A
OPA2197IDR
C45
100nF
C46
100nF
+/-10V
R51
1K
-2V5
D5
C431
C47
100nF
R52
40.2K
R53
10K
R54
0
U8B
OPA2197IDR
R55
10K
Output [0, 5V]
AnalogOut
Title Analog input
Project: Lock-in amplifier
Date: 26/07/2022
Revision: 1.0
Sheet 7 of 7
File: Analogin.SchDoc
Drawn by: Ignacio Horcas
UNED
ETS de Ingeniería Informática