Escuela Politécnica Superior de Linares

Grado en Tec. De Telecomunicaciones

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

INSTRUMENTO VIRTUAL BASADO EN ARDUINO, PARA LA MEDIDA Y CARACTERIZACIÓN DE CIRCUITOS LINEALES

Alumno: **Montoro Polo, José Mª**

Tutor: Reche López, Pedro J.
Depto.: Ingeniería de Telecomunicaciones

Junio, 2016

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

INSTRUMENTO VIRTUAL BASADO EN ARDUINO, PARA LA MEDIDA Y CARACTERIZACIÓN DE CIRCUITOS LINEALES

AGRADECIMIENTOS:

Gracias a mi mujer y a mi familia por su apoyo incondicional, y
a mis compañeros Miguel y Eva, por la ayuda en estos años
Sin todos vosotros no hubiera sido posible.

ABSTRACT

The project consists of making a virtual instrument to characterize lineal circuits by getting its frequency response. The virtual instrument is able to generate a sinewave that will be used to excite a lineal circuit. The frequency response of the circuit under test is computed by comparison between the output (system response) and the input. The virtual instrument is based on Arduino, and LabWindows/CVI is used for the software development.

RESUMEN

El proyecto ha consistido en la realización de un instrumento virtual, con el que poder caracterizar circuitos lineales, obteniendo su respuesta en frecuencia. El instrumento virtual es capaz de generar una onda sinusoidal de frecuencia variable, que excitará un circuito lineal. La respuesta en frecuencia del circuito a medir se calcula comparando la respuesta del sistema y la entrada. El instrumento virtual está basado en el uso de Arduino y se ha usado el entorno LabWindows/CVI para el desarrollo del software.

1. INTRODUCCION

En el último siglo el avance de la tecnología ha dado un salto enorme. La revolución microelectrónica en la industria, en el transporte, comunicaciones, hogares, ha hecho que la cantidad de datos que el ser humano ha de manipular en su vida diaria haya crecido exponencialmente a lo largo de los años.

El acercamiento de esta tecnología a los hogares y a personas de cada vez más temprana edad, hace que sea mayor la inquietud por aprender y descubrir cómo funcionan y que hacen los distintos aparatos que nos rodean, e incluso de fabricar nuestra propia tecnología.

Gracias al desarrollo de los ordenadores y el abaratamiento de los semiconductores, y con ello, de todo lo relacionado con la electrónica, son muchos los que investigan la forma de hacer su vida más fácil, mediante dispositivos electrónicos. Dichos dispositivos serían imposibles de desarrollar sin los instrumentos de medida necesarios para la caracterización de estos, y con ello, controlar y rectificar su funcionamiento. Los equipos de instrumentación que existen en el mercado están más orientados a nivel industrial, lo que hace imposible la adquisición por parte del "pequeño usuario" de dicha tecnología.

Luego, parece necesario el desarrollo de instrumentos de medida virtuales, que no dependan de un hardware extremadamente complejo, y que mantenga un precio razonable para un consumidor medio, permitiendo así un continuo desarrollo tecnológico.

1.1. INTRODUCCIÓN A LA INSTRUMENTACION VIRTUAL

Lo primero que se piensa cuando desconocemos algo es, ¿qué es?, ¿cuál es su origen? Una forma simple de definir un Instrumento Virtual sería la de "un instrumento que existe en función, pero no en una forma real" [1]. Esto no quiere decir que las señales que se midan no existan, sí que existen, pero de una forma diferente a la que se ve normalmente. Conociendo un poco de historia, se llegará a entender mejor qué son y de donde vienen los Instrumentos Virtuales.

Hace unos pocos cientos de años, la humanidad desconocía la electricidad. Entonces llegó el trabajo de Ohm, Oersted, Ampere, Watt, Maxwell y otros muchos. Ellos definieron el voltaje, la corriente y la potencia, naciendo así el campo de la electricidad. Este descubrimiento dio lugar a la distribución de la electricidad, iluminación de hogares, movimiento de máquinas y otras muchas capacidades. Esto hizo cada vez más

importante la medición de los parámetros eléctricos que se usaban. Durante la mayor parte del siglo XX, las medidas se concentraron en los parámetros eléctricos como el voltaje, corriente, potencia, factor de potencia, frecuencia, etc.

La edad de la electrónica comenzó con los tubos de vacío, radios, y televisores. Durante la II Guerra Mundial, la abundancia de la electrónica revolucionó sistemas de navegación, comunicaciones y el control de la mecánica y visual mediante la electricidad y electrónica [1].

Con el desarrollo del transistor, el mundo de la electrónica se expandió. Las limitaciones de tamaño y potencia disipada, empezaron a desaparecer. Los instrumentos de medición se hicieron más pequeños y consumían menos potencia.

Fue entonces cuando algunas compañías, como por ejemplo Fluke, Hewlett Packard, Tektronix, entre muchísimas otras, empezaron a desarrollar aparatos puramente de medida, desarrollando fuentes de potencia, sensores, traductores y pantallas. En la mayoría de los casos, los datos se tomaban y se pasaban físicamente a un datasheet, es decir, el uso de datos no era parte del paquete del instrumento de medida.

El campo del control industrial empezó a requerir más que simples mediciones de parámetros. Empezaron a desarrollarse sistemas de control que respondieran a los cambios de parámetros requeridos, mediante el uso de relés. Más tarde, se añadieron detectores e integrados, y empezaron a formarse sistemas de control PID.

Con el desarrollo del microprocesador, se hizo posible la unión de dos campos, el de instrumentación y computación. Se comenzaron a usar los ordenadores de forma muy específica, para determinadas aplicaciones, y, con el desarrollo de la microelectrónica y por tanto, la evolución de los ordenadores, se dio comienzo a la instrumentación virtual para actividades más complejas, que simplemente medir y tomar datos.

El costo de los aparatos de medida con gran capacidad de memoria, y que normalmente solo estaban disponible bajo pedido especial y para aplicaciones concretas, llevo a la pregunta de ¿cómo mejorar el rendimiento, sin aumentar el costo? La respuesta se encontró con los instrumentos virtuales. El desarrollo de ordenadores personales con gran capacidad de memoria hizo posible que los usuarios pudieran disponer de aparatos de medida muy especializados en su ordenador de sobremesa. El planteamiento de los fabricantes fue suministrar el software y el “hardware especial”, y que el usuario utilizase la potencia de cálculo de su propio ordenador.

Por tanto, un instrumento virtual se compone de algunas subunidades especializadas, un equipo de uso general, normalmente un ordenador, y un software.

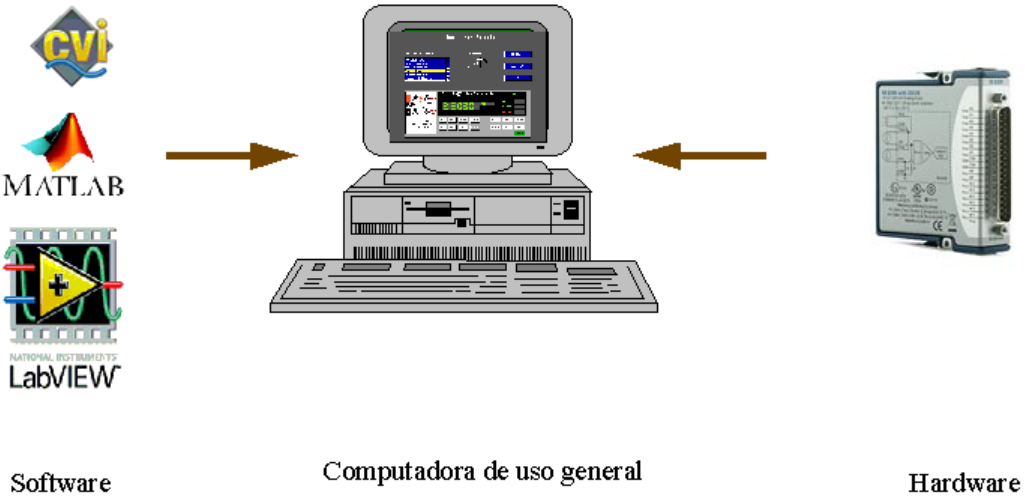


Figura 1: Instrumentación Virtual

A modo de ejemplo y desglosando brevemente el concepto del párrafo anterior, para que un instrumento haga una medición, sea de temperatura, humedad, etc., ha de disponer de un sensor. Ya que, probablemente, el parámetro que suministre el sensor no es eléctrico, hace falta un traductor para convertir dicho parámetro a una señal eléctrica. Tras obtener dicha señal, hay que acondicionarla para llevarla a niveles manipulables, pudiendo aplicarle amplificadores, filtros o rectificadores. Como último paso, debe haber un convertidor analógico-digital, para posteriormente, manipularlos. Una vez se tengan los datos en formato digital, ya se pueden procesar, almacenar, mezclar, comparar.

En la figura 2, se muestra el diagrama de bloques de lo anteriormente descrito pero adaptado a este proyecto en cuestión. Nótese, que el área rodeada en verde es la parte que, junto al software, proporcionaría el fabricante en un instrumento virtual comercializado. En este caso corresponde al hardware que se desarrollará en este proyecto.

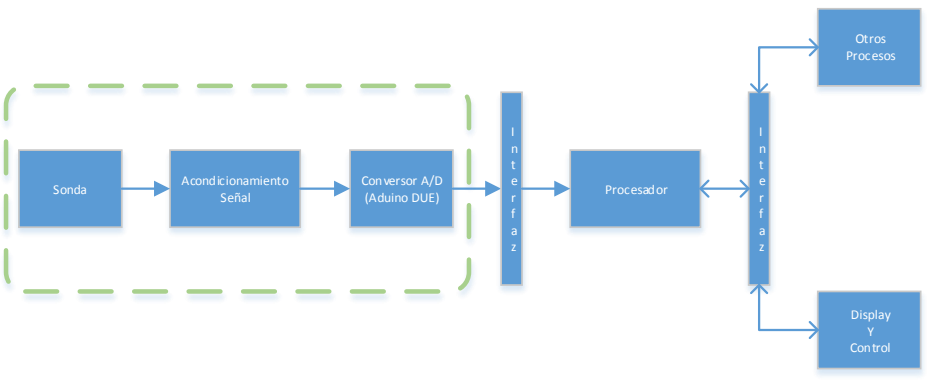


Figura 2: Diagrama de bloques de un Instrumento Virtual

1.2. PARTES DE UN INSTRUMENTO VIRTUAL

Como se ha visto en la introducción, un instrumento virtual abarca varias etapas. Las próximas secciones abarcarán las principales partes que componen la parte hardware del instrumento, y algunos de los software más 'genéricos' para controla dicho hardware.

1.2.1. SONDAS

La elección de una buena sonda es el primer paso para obtener una medida fiable de la señal. Las sondas están clasificadas en dos tipos: pasivas y activas. La diferencia radica en que las sondas activas necesitan de alimentación externa para activar los componentes activos que incluye la sonda en su interior, como pueden ser transistores y amplificadores. Estas sondas, además, proporcionan un ancho de banda mayor que las sondas pasivas.

1.2.1.1. SONDAS PASIVAS

La mayoría de aparatos de medidas utilizan este tipo de sondas. Son muy características en osciloscopios y en aparatos de medidas de frecuencia.

Dentro de las sondas pasivas, están las sondas de alta impedancia, y las sondas de baja impedancia. Las más utilizadas son las sondas de alta impedancia, cuya resistencia típica suele ser de 9MΩ, con lo que se consigue una relación de atenuación de 10:1 respecto a la entrada, siempre y cuando se conecte a una entrada de 1MΩ. Siendo así, la resistencia que se obtiene en el extremo de la punta es de 10MΩ. El esquema eléctrico de una sonda pasiva es el siguiente:

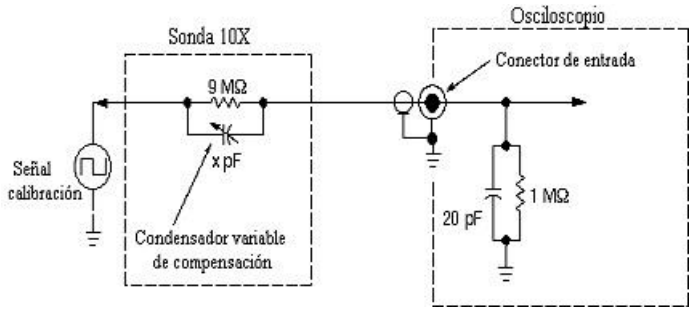


Figura 3: Esquema eléctrico de una sonda

Se puede observar en la figura anterior que se trata de un divisor de tensión. Aplicando la regla del divisor de tensión, comprobamos que, como bien se contaba antes,

la relación entre la tensión a la entrada del osciloscopio y la existente en la sonda viene dada por la ecuación 1.

$$V_{in} = V_{sonda} \frac{1M\Omega}{1M\Omega+9M\Omega} \qquad \textit{(ec. 1)}$$

Las sondas pasivas son sondas más robustas y baratas y además suelen poder medir tensiones mayores de 300V, pero, debido a la carga capacitiva que pueden imponer, ofrecen anchos de banda menores.

Por otro lado, están las sondas de baja impedancia. Estas sondas tienen una resistencia de entrada de 450Ω o 950Ω, con lo que se obtienen relaciones de atenuación de 10:1 o 20:1, contando con que la entrada del instrumento de medición sea de 50Ω. Las ventajas de este tipo de sondas son una baja carga capacitiva y un ancho de banda muy elevado, de entorno a 2GHz. En cuanto a las desventajas, al presentar una elevada resistencia, es posible que afecte a la amplitud de la señal que se quiera medir.

1.2.2. ACONDICIONAMIENTO DE LA SEÑAL

Se entiende por acondicionamiento de la señal el proceso de adaptación de la señal que queramos medir al instrumento de medida. Se puede amplificar o reducir la tensión, o la corriente. Por ejemplo, si la tensión de señal que se quiere medir, es del orden de mV, puede que el conversor A/D ni siquiera detecte dicha señal, por lo que será conveniente amplificarlos a unos valores adecuados para tal fin.

La pieza fundamental en un módulo de acondicionamiento de señal, es el amplificador operacional. Es un amplificador de alta ganancia, disponible en chip. Se compone de una entrada inversora (-) y una no inversora (+), alimentación positiva y negativa y una entrada para corregir las no idealidades del amplificador operacional.

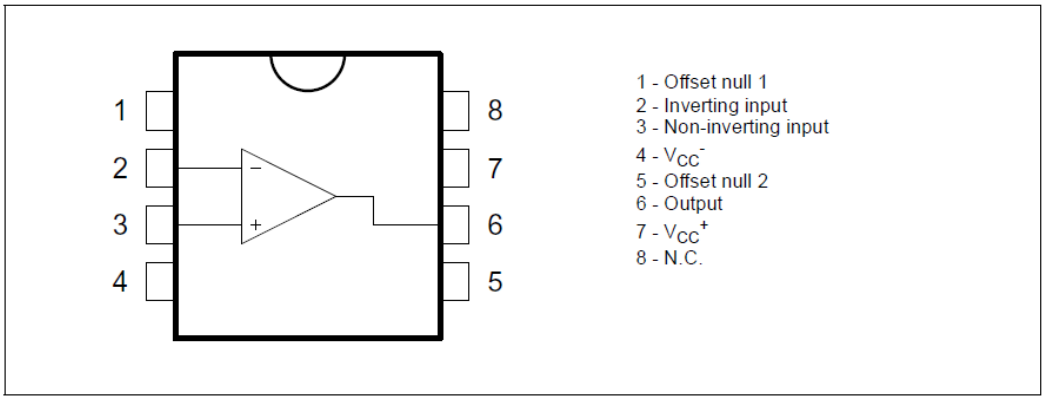


Figura 4: Amplificador operacional (Datasheet UA741CN)

Hay varias configuraciones disponibles para adaptación de señal, según el propósito que se desee conseguir. En la etapa de diseño del adaptador de señal, se estudiarán con más detalle algunas de ellas.

1.2.3. CONVERSOR ANALOGICO A DIGITAL (A/D)

Un conversor analógico-digital es un dispositivo capaz de convertir señales analógicas de voltaje en valores binarios. Los procesos que dan lugar a dicha conversión, son el muestreo, la cuantificación y la codificación.

El muestreo (del inglés Sampling), trata de tomar muestras periódicas de los valores de tensión de la señal analógica. La velocidad a la que se toman dichas muestras es la denominada frecuencia de muestreo, y viene dada en muestras por segundo. Para poder r7ecuperar la señal correctamente, la frecuencia de muestreo ha de ser, al menos, el doble de la frecuencia máxima de la señal ($F_s > 2F_{max}$). Esta condición viene dada por el teorema del muestreo de Nyquist –Shannon:

Teorema: Si la frecuencia más alta conocida en una señal analógica $x_a(t)$ es $F_{max} = B$ y la señal se muestrea a una velocidad $F_s > 2F_{max}$, entonces $x_a(t)$ se puede recuperar totalmente de sus muestras mediante la siguiente función de interpolación:

$$\boldsymbol{g(t) = \frac{sin(2\pi Bt)}{2\pi Bt}}$$

En el proceso de cuantificación, se transforman los valores de las muestras tomadas, en un conjunto finito de valores. El funcionamiento viene dado por dos parámetros: rango dinámico o intervalo de variación de la señal de entrada y el número de bits, que se relaciona directamente con el número de intervalos en que se divide el rango dinámico, con $nº\ div = 2^{nº\ bits}$. En el caso de este proyecto, Arduino cuenta con un conversor A/D de 12 bits, es decir, 2048 posibles valores [2].

Por último, en la codificación, se codifican o transforman las muestras cuantificado al código que se desee utilizar, siendo el más extendido el binario, aunque podría transformarse a otros códigos como BCD, Hexadecimal, Octal, etc. En Arduino, nos entrega la conversión directamente en decimal.

1.2.4. CONTROLADOR

El controlador es la parte encargada de sincronizar todas las operaciones del instrumento. Desde aquí se dan todas las órdenes, desde controlar al ADC, como de enviar la información al ordenador. Sus funciones principales en el instrumento serán.

- Recibir la información proporcionada desde memoria.
- Controlar la memoria.
- Enviar los datos al ordenador.

Debido a que nuestro controlador será la placa Arduino, la memoria y las ordenes a la memoria las hace internamente, aunque más adelante se verán lo inconvenientes de utilizar la memoria interna del propio Arduino.

1.3. INTRODUCCION A ARDUINO

"Arduino es una plataforma electrónica de código abierto basado en hardware y software fácil de usar. Está dirigido a cualquier persona que hace proyectos interactivos" (3).

Consiste en soporte hardware, formado por la alimentación, oscilador, y la carga del programa en microcontroladores de la serie de 8 bit de Atmel, aunque recientemente está expandiendo, y usando microcontroladores más potentes, como el SAM3X8E ARM Cortex-M3, también de Atmel, basado en tecnología de 32 bits. El software lo forman un entorno de desarrollo basado en Wiring [4], que es un framework usado para la programación de microcontroladores y un lenguaje, Processing [5], que está orientado al aprendizaje y realización de prototipo.

Arduino se presenta en su forma comercial en placas ya ensambladas, aunque, gracias a que es una plataforma libre, en la red se puede encontrar toda la información necesaria para el montaje de la placa por partes.

La plataforma presenta multitud de variantes en lo que respecta a placas con microcontrolador. La más famosa es Arduino Uno, que es la placa más extendida debido a que es la que lleva más tiempo en el mercado, y, gracias a la comunidad de Arduino, presenta muchas facilidades de uso debido a los miles de códigos que circulan por la red.

Otro tipo de placas son la Arduino Mega, Arduino Due, Arduino Mini, y Arduino Yún. Ésta última se distribuye con una versión libre de Linux, y tiene una mayor capacidad computacional que sus hermanas menores.

Para complementar la gran variedad de placas, se pueden encontrar en el mercado multitud de módulos de expansión, llamados Shield, con los que añadir funciones, como por ejemplo, tecnología GPS, WIFI, Bluetooh o GSM. También existe la posibilidad de fabricarse Shield personalizadas. Más adelante se explicarán los diferentes métodos existentes para crear estas Shield, ya que se implementarán en este proyecto.

1.3.1 HARDWARE

Como se ha mencionado anteriormente, una placa de Arduino más potente es la Arduino DUE [2], basado en un procesador de 32bits. Cuenta con 54 pines digitales de entrada/salida, de los cuales 12 permiten salidas PWM, y 12 entradas analógicas. Permite comunicación USB, con hasta cuatro puertos serial, tiene conector de alimentación externa, que conmuta automáticamente en caso de conectar ambos a la vez, para evitar daños en la placa. Como se observa, las características son bastante superiores que su hermana pequeña.

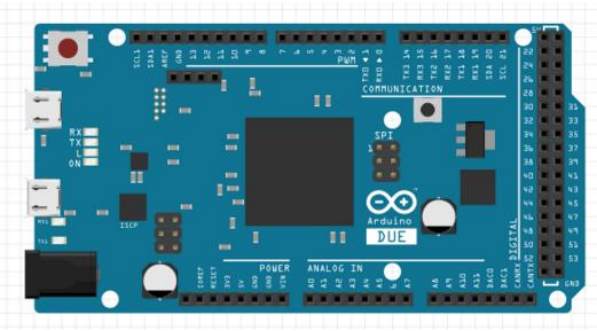


Figura 5: Esquema Arduino DUE

En la siguiente tabla, se muestran todas las características de la Arduino DUE Rev3:

Elemento	Valor
Microcontrolador	AT91SAM3X8E
Voltaje de Operación	3.3V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limites)	6-16V
Digital I/O Pines	54 (12 con PWM)
Entradas Analógicas Pines	12
Salidas Analógicas Pines	2 (DAC)
Total Salida de corriente DC en todas I/O	130 mA
Corriente DC 3.3V Pin	800 mA
Corriente DC 5V Pin	800 mA
Memoria Flash	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Velocidad de reloj	84 MHz
Largo de placa	101.52 mm
Ancho de placa	53.3 mm
Peso de placa	36 g

Tabla 1: Características Generales Arduino DUE

1.3.2. COMUNICACIÓN

Arduino ofrece la posibilidad de conectarse con el ordenador, o con otras Arduino u otros microcontroladores. Para la conexión con el ordenador ofrece una comunicación serie vía USB, que aparecerá en el ordenador como un puerto COM virtual.

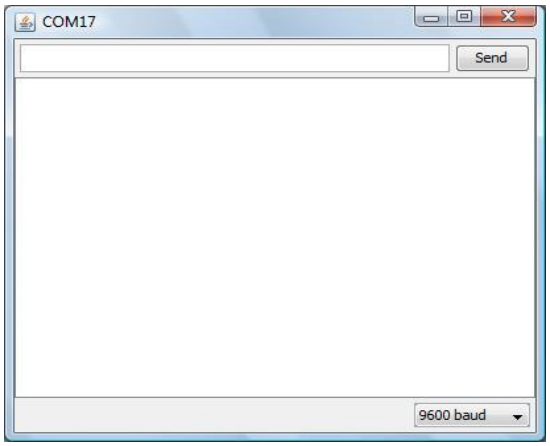


Figura 6: Terminal Arduino

Para la visualización de datos el propio software de Arduino presenta un terminal serie, donde se podrán visualizar datos simples que se envíen o reciban en la placa. Incluye leds SMD integrados para cerciorarse de que la comunicación se está realizando.

Aparte de la comunicación serial, también permite comunicación SPI.

1.3.3 SOFTWARE

El entorno de desarrollo de Arduino está basado en Wiring. Tiene un procesador de texto y un recuadro de mensajes.

Todos los programas de Arduino tienen un aspecto similar en su composición principal. Las dos funciones principales son “void setup()”, donde se configura la placa, por ejemplo, los pines, y la función “void loop()”, que es el bucle principal; esta función se ejecuta periódicamente ejecutando el código que lleva dentro. Como cualquier programa en C, también podemos incluir funciones personales.

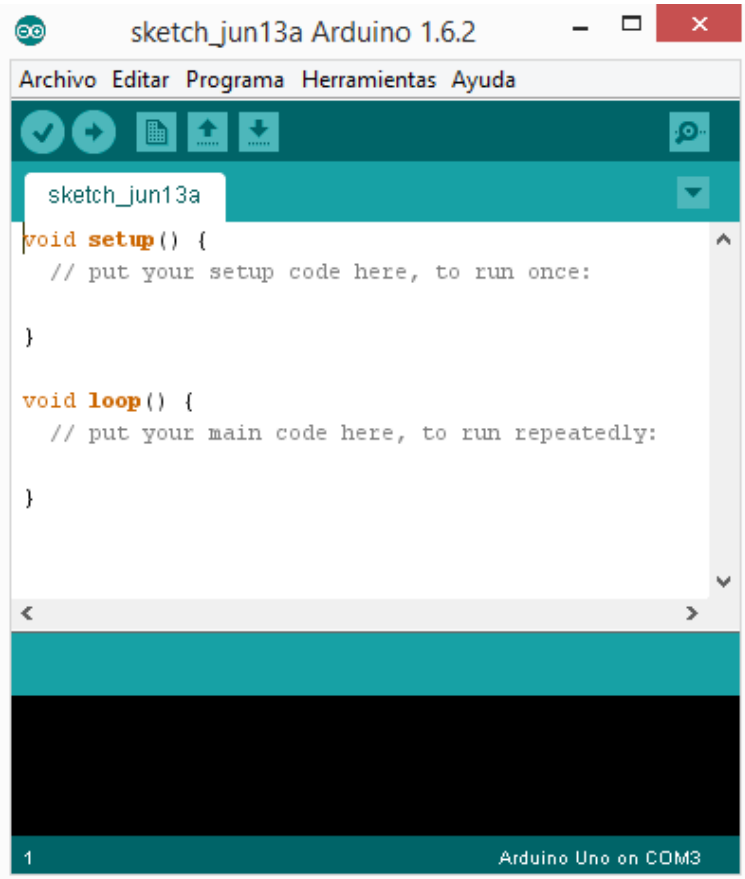


Figura 7: Entorno Arduino

Una vez se ha configurado el programa a desarrollar, y antes de compilar, se ha de seleccionar la placa de Arduino con la que se va a trabajar, y el puerto COM en la que se encuentra.

Cuando se inicia el entorno de desarrollo, en la parte superior tenemos una botonera de programación.

Figura 8: Botonera Arduino

En este caso se presenta la botonera de la versión 1.6.2 de Arduino. De izquierda a derecha, las funciones de los botones son las siguientes:

- Compilar: Verifica la sintaxis del código.
- Cargar: Envía el programa a la placa de Arduino. Como se mencionó anteriormente, ha de estar seleccionada correctamente el tipo de placa y el puerto al que está conectada.
- Nuevo: Crea un nuevo sketch. El sketch es la pantalla principal donde se escribe el código a desarrollar.
- Abrir: Muestra sketch ejemplos y los propios creados por el usuario, para poder cargarlos.
- Guardar: Guarda el trabajo realizado.
- Monitor serie: como se comentó en el apartado de comunicación, el monitor serie muestra los datos que se envían/reciben por el puerto USB.

1.4 CIRCUITOS LINEALES

Un circuito lineal es un ejemplo de sistema lineal e invariante en el tiempo (LTI por su acrónimo en inglés). La linealidad y la invariancia en el tiempo juegan un papel fundamental en el análisis de señales y sistemas. Esto es debido a que muchos procesos físicos poseen estas propiedades, por lo que pueden modelarse como sistemas LTI. Además, los sistemas LTI se pueden analizar con el suficiente detalle para proporcionar el conocimiento de sus propiedades.

Todos los sistemas LTI poseen la propiedad de la superposición, por lo que, si aplicamos a la entrada una combinación lineal de un conjunto de señales básicas, podemos utilizar la superposición para calcular la salida del sistema en términos de sus respuestas a estas señales básicas. Las señales muy generales se pueden representar

como la combinación lineal de impulsos retardados, que junto con las propiedades de superposición e invariancia en el tiempo permitirá caracterizar completamente un sistema LTI.

Pero, ¿cómo hallar la respuesta en frecuencia de un circuito? Suponiendo que a la entrada del circuito tenemos una señal continúa $x(t)$, la salida de éste viene dada por la convolución, notada con el símbolo *, en el tiempo de la entrada con la respuesta al impulso del circuito,

$$y(t) = x(t) * h(t) \qquad \textit{(ec. 2)}$$

En el domino transformado de Fourier la ecuación (2) queda como,

$$Y(\omega) = X(\omega) \cdot H(\omega) \qquad \textit{(ec. 3)}$$

Por lo que es fácilmente deducible, que la respuesta en frecuencia del sistema vendrá dada por la fracción de la señal de salida en el dominio transformado entre la señal de entrada en el mismo dominio.

1.4.1 TRANSFORMADA DISCRETA DE FOURIER

La Transformada Discreta de Fourier (DFT por sus siglas en inglés), juega un rol importante en el análisis, diseño e implementación de algoritmos y sistemas de señales en tiempo discreto. Las propiedades básicas de la transformada de Fourier propician el análisis de sistemas en el dominio de Fourier.

La DFT es idéntica a las muestras de la transforma de Fourier tomadas en frecuencias equiespaciadas. Por lo tanto, el coste computacional de una DFT de N muestras corresponde al coste computacional de N muestras de la transformada de Fourier en N frecuencias equiespaciadas $\omega_k = 2\pi k/N$. Hay algoritmos eficientes que son llamados algoritmos *fast Fourier transform* (FFT) para el cálculo eficiente de la DFT. Para obtener la máxima eficiencia, los algoritmos FFT deben ser calculados en los N valores de la DFT, por lo que si se desea calcular la DFT solo en una porción del rango de frecuencia $0 \leq \omega \leq 2\pi$, hay otros algoritmos que tal vez sean más flexibles, aunque son menos eficientes que el algoritmo de la FFT para el computo de todos los valores de la DFT. Uno de estos algoritmos, es el algoritmo de Goertzel, que es el que dará cabida en este proyecto.

Para el desarrollo de los siguientes capítulos, se ha de saber que la DFT, de una longitud finita de una secuencia de longitud N muestras es

$$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{kn}, \quad k = 0,1,\dots,N-1, \qquad \textit{(ec. 4)}$$

donde $W_N^{kn} = e^{-j\frac{2\pi}{N}}$.

En [6], se describe en mayor detalle el cálculo de la FFT

1.4.2 DIEZMADO EN EL TIEMPO ALGORITMO FFT

Los algoritmos de diezmado en el tiempo (8) están basados en descomponer la secuencia $x[n]$ en sucesivas sub-frecuencias más pequeñas.

El principio de los algoritmos diezmado en el tiempo se ilustra más considerablemente para el caso especial de que N sea un número entero potencia de 2.

Se puede separar la ecuación 4 $X[k]$ en dos secuencias de N/2 puntos, considerando puntos pares e impares

$$X[k] = \sum_{pares} x[n]\, W_N^{kn} + \sum_{impares} x[n]\, W_N^{kn} \qquad \textit{(ec. 5)}$$

$$X[k] = \sum_{r=0}^{\left(\frac{N}{2}\right)-1} x[2r]\, W_N^{2rk} + \sum_{r=0}^{\left(\frac{N}{2}\right)-1} x[2r+1]\, W_N^{(2r+1)k} \qquad \textit{(ec. 6)}$$

$$X[k] = \sum_{r=0}^{\left(\frac{N}{2}\right)-1} x[2r]\, (W_N^2)^{rk} + W_N^k \sum_{r=0}^{\left(\frac{N}{2}\right)-1} x[2r+1](W_N^2)^{rk} \qquad \textit{(ec. 7)}$$

Se puede demostrar que $W_N^{k2r} = W_{N/2}^{kr}$, por lo que se puede reescribir la ecuación 7 como

$$X[k] = \sum_{r=0}^{\left(\frac{N}{2}\right)-1} x[2r]\, W_{\frac{N}{2}}^{kr} + W_N^k \sum_{r=0}^{\left(\frac{N}{2}\right)-1} x[2r+1] W_{\frac{N}{2}}^{kr}$$

$$= G[k] + W_N^k H[k], \qquad k = 0,1,\dots,N-1 \qquad \textit{(ec. 8)}$$

Se puede continuar aplicando el diezmado a $G[k]$ y $H[k]$. El número de operaciones para la DFT_N es igual a las operaciones realizadas para dos $DFT_{N/2} + N$ productos, debido a los términos W_N^k.

Para el caso más general, se descompondrá hasta que se deje con una transformada de solo dos puntos (N=2). Esto requiere $v = log_2 N$ etapas de computación. Ya que en cada etapa tenemos N productos, la complejidad será $Nlog_2 N$.

La unidad computacional de la FFT se denomina mariposa (butterfly). Dicha estructura puede simplificarse sacando fuera el término W_N^r.

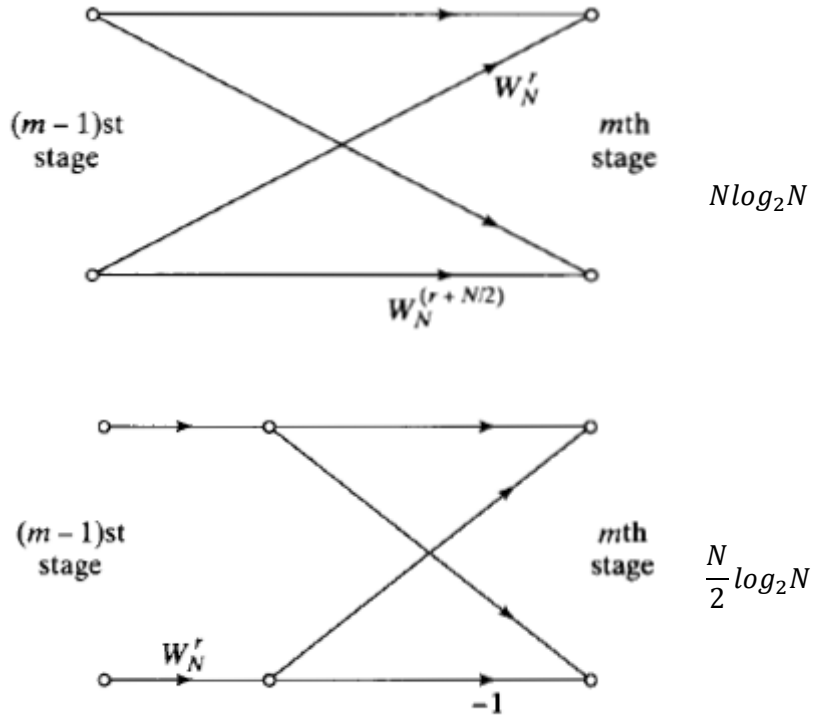


Figura 9: Comparación etapas mariposa genérica y simplificada

1.4.3 ALGORITMO DE GOERTZEL

El algoritmo de Goertzel [7], es un ejemplo de como la periodicidad de la secuencia W_N^{-kN} puede ser usada para reducir el coste computacional. Para derivar el algoritmo, se empieza notando que,

$$W_N^{-kN} = e^{j(2\pi/N)Nk} = e^{j2\pi k} = 1 \qquad \textit{(ec. 9)}$$

Esto es un resultado directo de la periodicidad de W_N^{-kN}. Se puede comprobar de forma que, al multiplicar la ecuación (9), por el lado derecho de la ecuación (4), no causa ningún efecto. Esto es

$$X[k] = W_N^{-kN} \sum_{n=0}^{N-1} x[n]\, W_N^{kr} = \sum_{n=0}^{N-1} x[n]\, W_N^{-k(N-r)} \qquad \textit{(ec. 10)}$$

Para poder definir el resultado final, se define la siguiente secuencia

$$y_k[n] = \sum_{r=0}^{N-1} x[r]\, W_N^{-k(n-r)} u[n-r] \qquad \textit{(ec. 11)}$$

De la ecuación 10 y 11 y dando por hecho que $x[n] = 0$ para y $n \geq N$, se sigue que

$$X[k] = y_k[n]|_{N=n} \qquad \textit{(ec. 12)}$$

La ecuación 7 se puede interpretar como una convolución discreta de la secuencia de duración finita $x[n]$,$0 \leq n \leq N-1$ con la secuencia $W_N^{-kN} u[n]$. Por lo tanto, $y_k[n]$ puede ser vista como la respuesta de un sistema con respuesta al impulso $W_N^{-kN} u[n]$, a una entrada de longitud finita $x[n]$. En particular, $X[k]$ es el valor de la salida cuando n=N.

El algoritmo de Goertzel es atractivo cuando N es más pequeña $log_2 N$, pero cuando hay que calcular todos los valores de N, este método no es el más indicado, ya que su coste computacional es N^2. Se puede ver un análisis más detallado [6]

1.4.4 FILTROS ACTIVOS

Dado que uno de los objetivos de este proyecto es la creación de varias etapas de filtros activos, se va a proceder a dar una pequeña introducción de estos, para que más adelante se puedan implementar distintos modelos.

Entonces, ¿Qué es un filtro? Se puede definir como un sistema que permite el paso de señales eléctricas única y exclusivamente a un rango determinado de frecuencias, impidiendo el paso del resto.

Son utilizados para el acondicionamiento de señales, de entrada o de salida, digitalización, selectores de frecuencia etc.

En función de su función de transferencia se clasifican en Paso Bajo, Paso Alto, Paso Banda, y Elimina Banda. También podemos clasificarlos por su tecnología o en función al tipo de implementación.

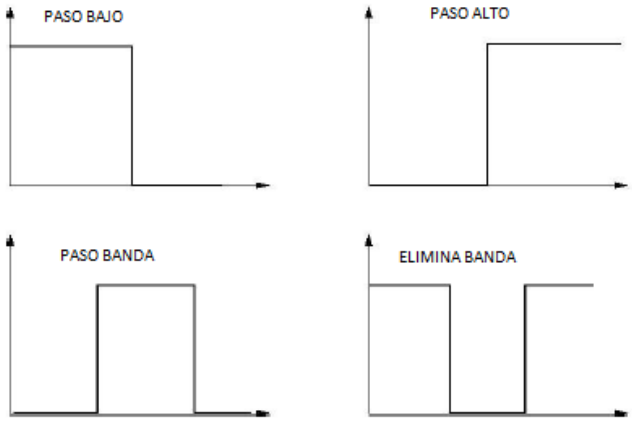


Figura 10: Filtros ideales según su función de transferencia

Al contrario de los ideales, los filtros reales tienen los siguientes defectos que los puede perjudicar:

- La transición entre banda de paso y la banda a eliminar, no es inmediata, si no que depende del orden del filtro.
- La respuesta en fase no es lineal, por lo que aumenta la distorsión de la señal.

Por lo general, se calcula la ganancia y fase del filtro siguiendo uno de los siguientes criterios:

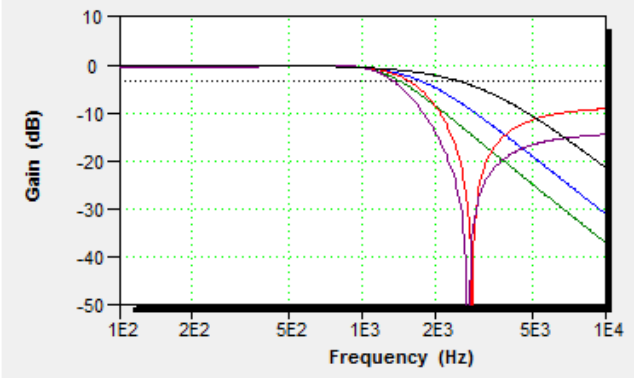
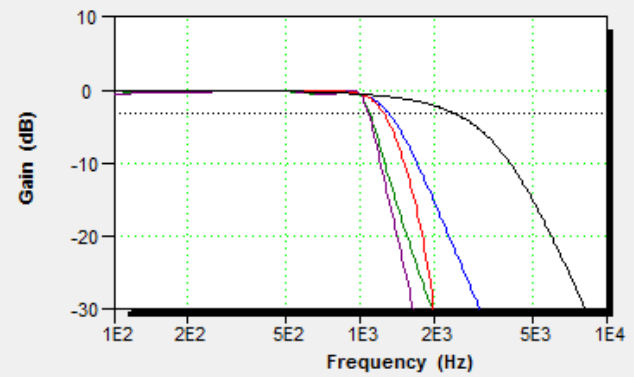
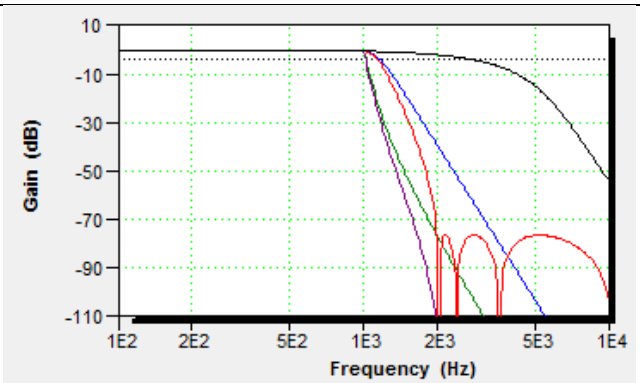
- Respuesta máxima plana en banda de paso
- Transición rápida entre banda de paso y banda de parada.
- Respuesta de fase lineal.

Para ello, la función de transferencia deberá tener polos complejos.

$$F(s) = \frac{A_0}{(1+a_1.s+b_1.s^2)+(1+a_2.s+b_2.s)+\cdots(1+a_n.s+b_n.s^2)} \qquad \textit{(ec. 13)}$$

Partiendo de esta ecuación, los filtros que se pueden realizar son el Butteworth, que optimiza la respuesta plana de la banda de paso, el Chebychev, que tiene una respuesta mucho más abrupta, pudiendo optimizar la velocidad de transición, y el filtro de Bessel, que optimiza la respuesta en fase.

La tabla siguiente muestra una comparativa entre los distintos filtros que existen, y con diferente orden. Cada uno de ellos sitúa la frecuencia de corte en 1KHz.

Orden N=2		LEYENDA Butterworth Chebyshev Inverse Chebyshev Elliptic (Cauer) Bessel (Thompson)
Orden N=4		LEYENDA Butterworth Chebyshev Inverse Chebyshev Elliptic (Cauer) Bessel (Thompson)
Orden N=8		LEYENDA Butterworth Chebyshev Inverse Chebyshev Elliptic (Cauer) Bessel (Thompson)

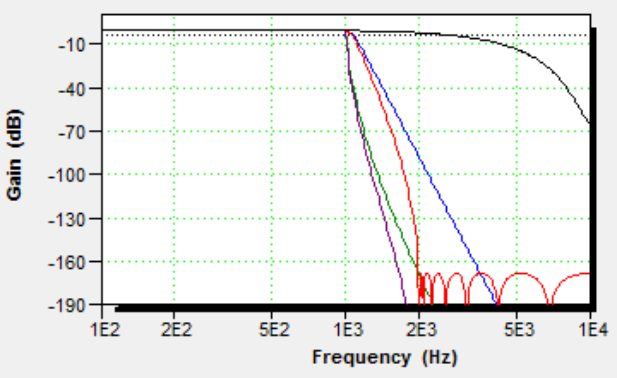
Orden N=16		LEYENDA Butterworth Chebyshev Inverse Chebyshev Elliptic (Cauer) Bessel (Thompson)

Tabla 2: Comparación de filtros respecto a su orden

Puede observarse que a mayor orden, mejoran mucho las características de los filtros, aunque la complejidad aumenta demasiado, por lo que se ha de buscar un equilibrio entre la eficiencia del filtro y la complejidad de montaje.

Por otra parte, se puede diseñar el filtro en función de su factor de calidad, en vez de en el orden del mismo. En filtros pasa banda, el factor de calidad se define como:

$$Q = \frac{f_m}{(f_2 - f_1)} \qquad \textit{(ec. 14)}$$

Siendo f_m la frecuencia central del filtro, y f_1 y f_2 las frecuencia de corte inferior y superior respectivamente. En filtro pasa-bajo, o pasa-alto, el factor de calidad queda definido como:

$$Q = \frac{\sqrt{b_i}}{a_i} \qquad \textit{(ec. 15)}$$

Gráficamente, los valores de Q se pueden calcular como la distancia entre la línea de 0dB y el punto pico de la respuesta al filtro.

2. OBJETIVOS

El objeto principal de este proyecto es el desarrollo de un instrumento virtual de medida basado en Arduino. El instrumento virtual desarrollado debe permitir, al menos, obtener la respuesta en frecuencia de circuitos lineales. Una vez desarrollado, se comprobará el funcionamiento de dicho instrumento mediante la implementación de varios filtros activos con diferentes respuestas en frecuencia, paso bajo, paso alto y paso banda.

La metodología a desarrollar para el cumplimiento de los objetivos anteriores se presenta a continuación:

1. Estudio y elección de las placas de Arduino para desarrollar.
2. Elección de los componentes externos que completen la parte hardware junto a Arduino.
3. Elección del software elegido para la implementación del instrumento virtual, a elegir entre LabWindow/CVI o Matlab.
4. Desarrollo de hardware y prueba en placa protoboard.
5. Desarrollo del software elegido en puntos anteriores.
6. Diseño de varios circuitos lineales, en este caso tres filtros activos de diferentes características.
7. Prueba de los circuitos anteriores con instrumento externo para comprobar su correcto funcionamiento, y desarrollo de estos en placa perforada o placa de circuitería.
8. Prueba de los circuitos anteriores con el instrumento virtual.
9. Desarrollo en placa perforada o de circuito impreso de hardware necesario para el instrumento virtual.
10. Presentación de resultados.

3. ESTADO DEL ARTE

En este apartado se estudiará la actualidad en las tecnologías de instrumentación virtual, y de los diferentes instrumentos que hay.

Hay una gran variedad de instrumentos virtuales, desarrollados tanto por grandes compañías como por aficionados a la electrónica. La variedad de productos es muy grande, pero hay que tener en cuenta que son productos cerrados, es decir, no es posible incorporar más prestaciones a dicho instrumento. También se ha de mirar el costo de dichos instrumentos, que, por lo general, no suele ser pequeños.

Debido a que la instrumentación virtual está basada en software, si puede digitalizarse, puede medirse. Por lo tanto, el hardware de medición puede verse en dos ejes, resolución (bits) y frecuencia. A continuación se muestra una imagen con la comparación las capacidades de medición de hardware de instrumentación virtual con instrumentación tradicional a lo largo del tiempo. Para información más detallada consultar [8].

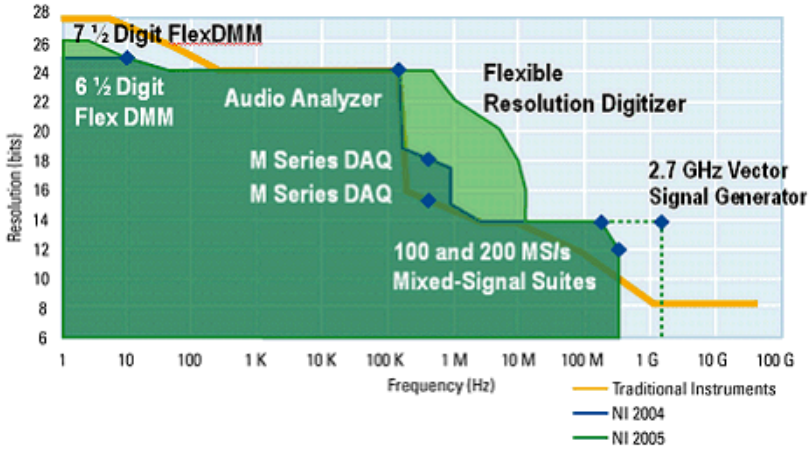


Figura 11: Inst. Virtual vs. Inst. Tradicional

La gráfica anterior está referida a instrumentación de la compañía National Instrument (NI), referencia mundial en instrumentación virtual.

En la actualidad, con la tecnología PLD (Programmable Logic Device), es posible realizar instrumentos más versátiles. En 2006, se desarrolló un interesante proyecto basado en el diseño de un sistema denominado “Instrumentación Virtual Reconfigurable (RVI)”. El RVI propone un dispositivo conectado a un PC a través de algún puerto (USB, Ethernet, etc.). Una aplicación de software de alto nivel despliega una lista de instrumentos y permite configurar el sistema para convertirse en dicho instrumento.

La tendencia básica en la industria de las pruebas automatizadas es un cambio muy marcado hacia sistemas de pruebas basados en software. Por ejemplo, el Departamento de Defensa de los EE.UU. ha especificado que en los futuros sistemas ATE (Equipo para Pruebas Automatizadas), deben seguir una estructura modular de hardware y software reconfigurable, llamada instrumentación sintética. Este tipo de instrumentación refleja el claro cambio que deja a un lado el hardware como pieza fundamental del sistema, y se centra más en el software.

En NI, el Grupo de Trabajo de Instrumentos Sintéticos, define un instrumento sintético como “un sistema reconfigurable que conecta una serie de elementos de hardware y componentes de software con interfaces estándar para generar señales o realizar mediciones utilizando técnica de procesamiento numérico” [8].

4. MATERIALES Y MÉTODOS

A continuación se presentan las diferentes etapas que se han utilizado, así como la descripción de su funcionamiento y el proceso hasta completar un instrumento capaz de cumplir los objetivos de la sección 2.

4.1. DESCRIPCIÓN DEL PROYECTO

Como se observa, se divide la implementación del proyecto en cuatro etapas. Se puede observar que no todo el hardware se implementa a la vez. Esto se debe a que el desarrollo del proyecto es en serie, es decir, cada etapa depende de la inmediata anterior.

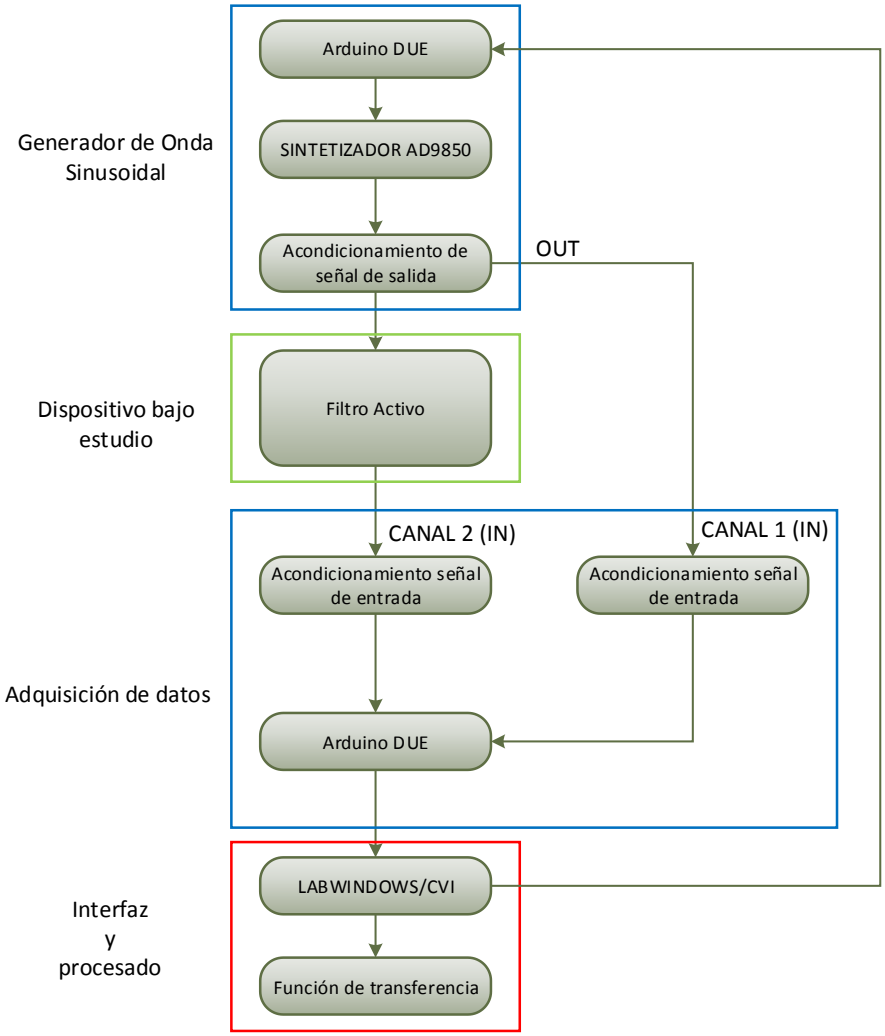


Figura 12: Diagrama completo instrumento virtual

Explicando brevemente cada una de las etapas del diagrama de la figura 13 se tiene:

Primera etapa. Se genera una seña sinusoidal mediante un sintetizador DDS (Síntesis Digital Directa), utilizando Arduino como controlador de dicho sintetizador, y acondicionando la señal a unos valores prácticos de funcionamiento.

Segunda etapa. Se desarrollan los diferentes circuitos de prueba, incluyendo cálculo, diseño y montaje en placa protoboard.

Tercera etapa. Aquí se implementa la parte de adquisición de datos. Se ha de acondicionar las entradas de señal de ambos canales.

Cuarta etapa. En esta etapa se analizan los dos programas objeto del proyecto, y se decide por cuál de los dos usar. Se crea la Interfaz del instrumento virtual, y se procesa toda la información recibida.

Hay que destacar que se ha de hacer hincapié en las diferentes etapas de adaptación de señal, con el fin de obtener las medidas lo más limpias posibles,

Para la prueba de las diferentes etapas, se hará uso de un osciloscopio Hantek6022BE como apoyo al proyecto. Con todas estas etapas se cerraría el módulo propuesto para este Trabajo Fin de Grado**.**

4.2. ETAPA 1: GENERADOR DE ONDA SINUOIDAL

Como se comentó en el avance, para la generación de señales se ha implementado un sintetizador DDS.

El AD9850 [9] es un sintetizador de Analog Device que usa Síntesis Digital Directa para generar un seno mediante el control de un oscilador numéricamente. Esa señal seno se convierte a analógica mediante un conversor Digital a Analógico interno de alta velocidad.

En la siguiente figura se muestra el diagrama de bloques y la señal de flujo:

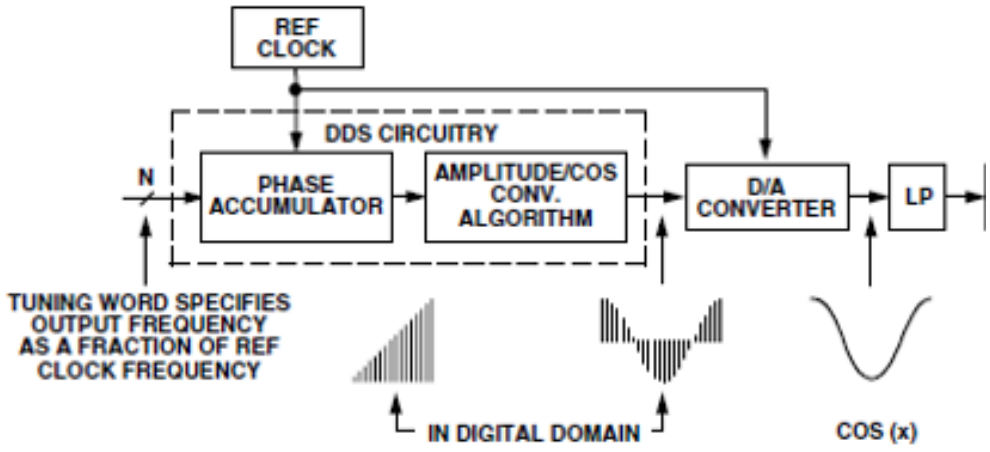


Figura 13: Diagrama de bloques básico DDS y flujo de señal de AD9850

El esquema eléctrico que proporciona dicho funcionamiento se consigue configurando el AD9850 de la siguiente manera.

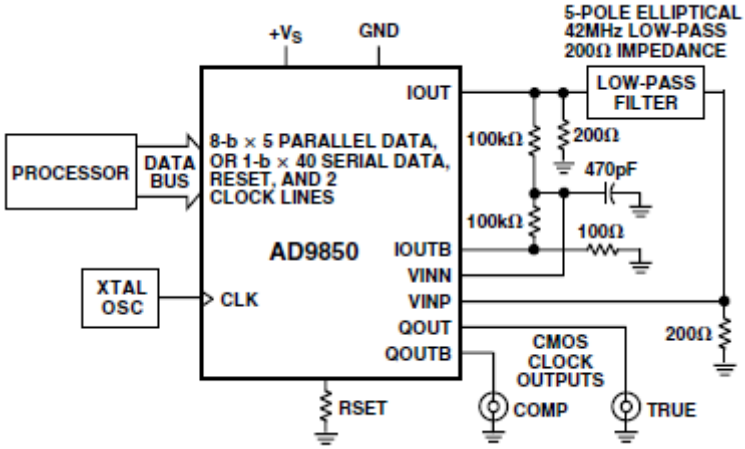


Figura 14: Generador con filtro paso bajo

En este proyecto, el AD9850 ya tiene la configuración montada, ya que, aparte del encarecimiento del producto por separado, el montaje del AD9850 es bastante complejo, debido a las pequeñas dimensiones de este.

Dispone de un registro de entrada externa de datos de 40 bits, que pueden cargarse en paralelo o en serie, le sigue otro registro de datos interno que al mismo tiempo envía a la DDS una palabra de 32 bits que contiene el valor de la frecuencia y otra de 8 bits para modulación de fase, además de otros controles internos. Los datos de fase frecuencia y control gobiernan al DDS, la cual comanda un convertidor digital a analógico de alta velocidad de 10 bits. Más detalles del sintetizador en [9].

Dispone de una frecuencia de reloj muy precisa de 125MHz (Vcc = 5v) o 110MHz (Vcc = 3.3v), por lo que se pueden generar señales senoidales de gran pureza espectral. La frecuencia puede ser cambiada digitalmente, hasta 23 millones de frecuencias cada segundo.

Las palabras de sintonía, fase y control pueden cargarse en el interior del chip de forma serie o paralelo. En paralelo se manda 5 palabras de 8 bits en los pines D0-D7, mientras que en serie se mandan 40bits hacia el terminal de entrada D7.

El sintetizador elegido, aunque hay varios por la red ha sido el siguiente:

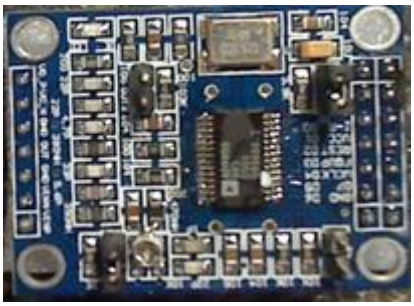


Figura 15: AD9850

A continuación se detallan las patillas del circuito, para posteriormente entrar a su programación.

D0-D7	Bus de programación paralela
GND	Tierra
WCLK	Reloj de programación serie
FQUP	Pestillo serie de programación
D7	Programación serie. Mismo pin que D7
REST	Reestablecer
SQW	Salida de onda cuadrada
sinA	Salida senoidal sin filtrar
sinB	Salida senoidal filtrada. Filtro PB 70MHz

Tabla 3: Pines AD9850

En este proyecto se ha implementado el funcionamiento en serie, ya que resulta más sencillo de controlar y se utilizan menos pines de la placa Arduino.

A continuación se muestra el esquema eléctrico con el conexionado del sintetizador y Arduino. Para este montaje solo se han utilizado los pines digitales, ya que solo es necesario controlar las entradas del AD9850, con niveles altos, o bajos.

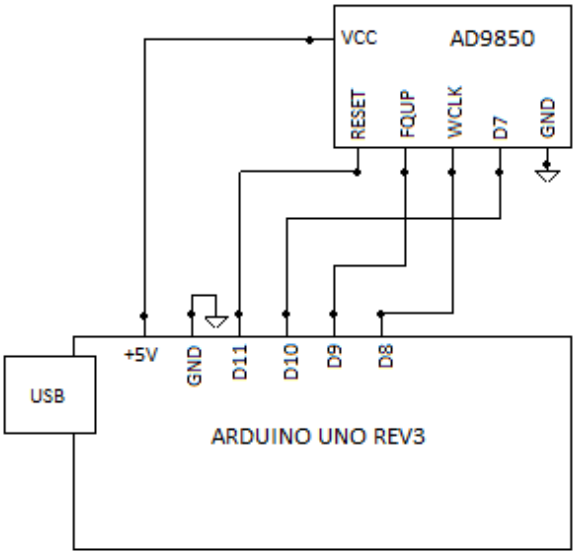


Figura 16: Conexionado AD9850 con ARDUINO DUE

La alimentación para el sintetizador se obtiene de la propia alimentación que suministra Arduino. La potencia disipada, alimentada a 5V, es de 380mW, por lo que tenemos un consumo de corriente de:

$$I = \frac{380mW}{5V} = 76mA \qquad \textit{(ec. 16)}$$

La corriente necesaria para el funcionamiento de las entradas de programación, según datasheet, son de $12\mu A$, por lo que se pueden alimentar correctamente con las salidas digitales de Arduino, que suministran un máximo de 40mA.

Para el cálculo de la palabra de sintonización, se ha de seguir la siguiente fórmula, que puede encontrarse en el *datasheet* del AD9850:

$$\Delta Fase = \frac{f_{out} \cdot 2^{32}}{CLKIN} \qquad \textit{(ec. 17)}$$

Donde,

∆Fase es el valor de la palabra de sintonización de 32 bits.

CLKIN es la frecuencia del reloj expresada en MHz.

F_{OUT} es la frecuencia de la señal de salida en MHz.

Una vez conectado todo, se puede comenzar a programar la placa Arduino. Para la programación, es de gran ayuda el datasheet del fabricante, que tiene un apartado especialmente para la programación del AD9850.

Como se dijo anteriormente, se usara el modo serie para ahorro de pines en Arduino. En el modo de carga serie, los flancos de subida de WCLK cambian los datos de 1 bit en el pin D7, a través de los 40 bits de información de programación.

Investigando un poco, la cuestión era ¿Cómo enviar los datos en serie desde Arduino? La respuesta llegó desde la propia página de Arduino, y de la gran comunidad que la asiste. Para realizar este desplazamiento, se usa la función *shiftOut [10]*. La función *shiftOut*, desplaza 1 byte de datos de bit en bit. Puede comenzar por el bit más significativo, o por el bit menos significativo. La sintaxis de la función en Arduino es la siguiente:

Shiftout(pinDatos, pinReloj, bitOrder, valor)

pinDatos: pin de salida de cada bit.
pinReloj: cambia una vez que el *pinDatos* se ha establecido en un valor correcto.
bitOrder: forma en que cambian los bits; MSBFIRST, bit más significativo primero, LSBFIRST, bit menos significativo primero.

Por lo tanto, solo se tiene que calcular la palabra de sintonización mediante la ecuación (17), y posteriormente mandarla bit a bit, hacia la entrada D7, para almacenarla en el registro de 40 bits. Primero se calcula la palabra de sintonización con el dato que se

le introduce desde el puerto serie, y a continuación manda la palabra de 32 bits, empezando por el bit menos significativo.

Es importante, y no se puede olvidar, poner todas las entradas a cero antes de empezar a mandar cualquier información, por lo que en el código se implementan dos funciones para inicializar a cero el dispositivo, AD9850_INICIO y AD9850_RESET borrando así cualquier rastro de información que haya en los registros.

Los resultados se podrán comprobar el apartado de resultados, descrito más adelante.

Tras la configuración del sintetizador, se ha de adecuar la señal de salida. El sintetizador, aunque sintetice una señal sinusoidal perfecta, no genera señales negativas. Para ello se monta una etapa de salida, compuesta por un amplificador operacional, con el cual cambiaremos la tensión de referencia de la señal, situando el valor de su voltaje medio en cero, y amplificándola al doble de la tensión. En este caso, la señal de salida tendría 2Vpp con paso medio por cero. El esquema de dicha etapa se muestras en el ANEXO III: ESQUEMÁTICO DE CIRCUITOS, de este proyecto.

4.3. ETAPA 2: ADQUISICION DE DATOS

Este apartado se centra en el desarrollo del hardware necesario para la adquisición de datos. Para ello se recurre a Arduino DUE, que incluye un ADC con una resolución de 12 bits y 1MS/s. Para llegar a esta velocidad de muestreo, es necesario modificar los registros del microcontrolador ARM, debido a que los valores de fábrica vienen muy por debajo de 1MS/s. Según datasheet de fabricante, no es recomendable forzar al Arduino a esta velocidad, por lo que, como se verá más adelante, la velocidad máxima será de 666KS/s. Esto se consigue ejecutando el reloj del ADC en modo libre, es decir, ejecutándose libre de una manera continua.

Para realizar la tarea de aumentar la velocidad de muestreo, lo primero es recurrir a la tabla del mapeado de pines de Arduino DUE [11]. En ella se han de localizar los pines que corresponden a las entradas analógicas de Arduino DUE A0 y A1, que son las entradas que se utilizarán para este proyecto.

Due Pin Number	SAM3X Pin Name	Mapped Pin Name	Max Output Current (mA)	Max Current Sink (mA)
54	PA16	Analog In 0	3	6

55	PA24	Analog In 1	3	6
56	PA23	Analog In 2	3	6
57	PA22	Analog In 3	3	6
58	PA6	Analog In 4	3	6
59	PA4	Analog In 5	3	6
60	PA3	Analog In 6	3	6
61	PA2	Analog In 7	3	6
62	PB17	Analog In 8	3	6
63	PB18	Analog In 9	3	6
64	PB19	Analog In 10	3	6
65	PB20	Analog In 11	3	6

Figura 17: Mapeo Pines SAM3X / Arduino DUE

Como se observa en la tabla anterior, los pines del SAM3X correspondientes a las entradas analógicas elegidas, son la PA16 y PA24. Con esta información, se ha de buscar en el datasheet del controlador a que pines del ADC pertenece.

Instance	Signal	I/O Line	Peripheral
ADC	ADTRG	PA11	B
ADC	AD0	PA2	X1
ADC	AD1/WKUP1	PA3	X1
ADC	AD2	PA4	X1
ADC	AD3	PA6	X1
ADC	AD4	PA22	X1
ADC	AD5	PA23	X1
ADC	AD6	PA24	X1
ADC	AD7	PA16	X1
ADC	AD8	PB12	X1
ADC	AD9	PB13	X1
ADC	AD10	PB17	X1
ADC	AD11	PB18	X1
ADC	AD12	PB19	X1
ADC	AD13	PB20	X1
ADC	AD14/WKUP13	PB21	X1

Figura 18: Mapeo pines ADC del SAM3X

Según esta tabla, las entradas analógicas del Arduino DUE, A0 y A1, corresponden a las entradas AD7 y AD6, respectivamente, del conversor ADC del

microcontrolador SAM3X. Una vez conocidas las entradas del ADC, se pueden modificar los registros del microcontrolador SAM3X. Todo ello, es programado desde el entorno de Arduino.

Para modificar los registros, es necesario recurrir al datasheet del SAM3X. Los principales registros a modificar para la configuración inicial del ADC son: **ADC Control Register** (ADC_CR), utilizado para reinicializar y dar comienzo a la conversión, el **ADC Mode Register** (ADC_MR), donde se realizan varias acciones, como configurar el prescaler, poner el ADC en modo libre, etc., y el **ADC Channel Enable Register** (ADC_CHER), usado para activar o desactivar los canales del ADC. Al tratarse de un microcontrolador de 32 bits, los registros son del tipo 0x00000000 (en código hexadecimal). Todos los valores que se muestran a continuación, se han de modificar en el 'setup' de Arduino.

Primero se ha de configurar el **Mode Register**. La siguiente tabla muestra la estructura de dicho registro.

31	30	29	28	27	26	25	24
USEQ	–	TRANSFER		TRACKTIM			
23	22	21	20	19	18	17	16
ANACH	–	SETTLING		STARTUP			
15	14	13	12	11	10	9	8
PRESCAL							
7	6	5	4	3	2	1	0
FREERUN	FWUP	SLEEP	LOWRES	TRGSEL			TRGEN

Figura 19: Mode Register ADC SAM3X

Para más información de lo que modifica cada bit, consultar páginas 1333 a 1335 de [12].

En este caso, para poner el ADC en modo libre, se ha de poner el bit 7 a "1". Para ello, se ha modificar la primera línea de la forma 0x80. De esta manera se pone el bit 7 a "1" y todos los demás a "0".

Tras modificar el **Mode Register**, se han de habilitar los canales, en este caso el AD7 y el AD6. Para ello vemos como está compuesto el **Channel Enable Register**.

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

Figura 20: Channel Enable Register ADC SAM3X

Se han de activar los bits 7 y 6 de este registro. Para ello se modifica de la forma 0xC0, que corresponde al número binario ‘1100000’.

Tras activar los canales, lo único que falta es activar el ADC, con el **Control Register**. La configuración de dicho registro está compuesta como.

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	START	SWRST

Figura 21: Control Register ADC SAM3X

En este caso, el registro solo consta de 2 bits. Por tanto se ha de poner el bit 2 del registro a “1”. Para ello, se ha de modificar para escribir en él el número binario ‘10’, correspondiente al número decimal 2.

Una vez configurado, el siguiente paso consiste en coger los datos del ADC. Para ello, las siguientes configuraciones se realizan en el ‘loop’ de Arduino.

Antes de leer los datos, se ha de esperar a que la conversión del ADC haya terminado, por lo que se ha de leer el **Interrupt Status Register.**

31	30	29	28	27	26	25	24
–	–	–	RXBUFF	ENDRX	COMPE	GOVRE	DRDY
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
EOC15	EOC14	EOC13	EOC12	EOC11	EOC10	EOC9	EOC8
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

Figura 22: Interrupt Status Register ADC SAM3X

En este caso, los bits que se han de leer son el bit 7 y 6. Con un ‘0’, la conversión no ha finalizado; con el bit en ‘1’, el canal está habilitado y la conversión finalizado.

Para leer los datos, es necesario acceder al **Channel Data Register**. El registro se muestra en el datasheet de la siguiente manera.

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	DATA			
7	6	5	4	3	2	1	0
DATA							

Figura 23: Channel Data Register ADC SAM3X

Para la configuración de este registro, únicamente se indica el bit en el que está guardado el dato. Los bits están ordenados respecto a los canales del ADC, por lo que, para obtener los datos, será necesario leer los datos del bit 7 y 6, correspondientes a los canales analógicos A0 y A1, almacenándolos en una variable.

Tras la configuración del software, es necesario la implementación de una etapa para el acondicionamiento de la señal que entra al Arduino, para cada uno de los canales. Arduino solo admite voltajes positivos a su entrada, por lo que, para acondicionar la señal, se ha utilizado un amplificador operacional sin ganancia. De esta forma, se le aplica una corrección de voltaje tal que todas las muestras sean positivos, y dicha corrección se anulará mediante software en el propio código de Arduino. Ya que la señal de entrada máxima que admite Arduino es de 3,3v, a las etapas de entrada no se le asignará ninguna ganancia, dejando así un margen de 1.3v.

4.4. ETAPA 3: IMPLEMENTACION DEL SOFTWARE

Para implementar el software, en este proyecto se ha decantado por LABWINDOWS/CVI de National Instrument [13]. Es un software dedicado por completo a la adquisición y procesamiento de datos. Está basado en ANSI C, e incluye cientos de librerías relacionadas con el procesado de señales.

El programa está basado en Multithreading, que es una forma de multiprocesado. El sistema operativo ejecuta cada tarea por separado durante un periodo de tiempo de forma que parece que todas las tareas se están ejecutando simultáneamente. Con esto se consigue fluidez en la ejecución del programa, sobre todo en la interfaz. Para realizar el Multithreading se han utilizado funciones ya implementadas de LABWINDOWS. Para completar información y ver ejemplos visitar el documento web [14].

El programa está separado en 5 procesos separados. Dichos procesos están encargados de lo siguiente:

- Proceso1: Es el proceso principal del programa. Sobre todo se encarga de ejecutar la interfaz de usuario y pequeñas tareas que tenga un pequeño coste computacional.
- Proceso 2: Encargado de realizar la lectura de los datos y almacenarlos en el PC.
- Proceso 3: Procesa los datos al mismo tiempo que se van recibiendo en el proceso anterior. En este proceso hallamos la frecuencia de entrada de la señal así como otros parámetros.
- Proceso 4: Realización de la respuesta en frecuencia mediante la búsqueda de máximos en la FFT.
- Proceso 5: Realización de la respuesta en frecuencia mediante el algoritmo de Goertzel.

El instrumento virtual dispone de varias herramientas, de manera que se pueda llevar a cabo un análisis lo más completo posible de los circuitos que con él se analicen.

La pantalla principal se ha diseñado con la apariencia tal de un osciloscopio. En ella se podrá observar la señal en tiempo real (con retardo, debido a la transmisión vía USB).

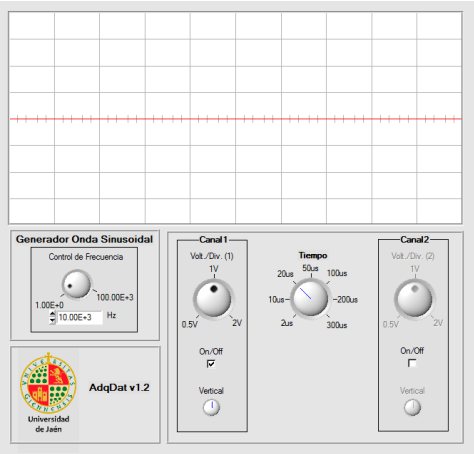


Figura 24: Instrumento virtual 1

En ella se pueden ver los controles típicos de un osciloscopio, pero donde además se ha añadido el control para el manejo del generador de frecuencia.

Para el análisis y cálculo de la señal, como bien se ha dicho antes, se ha separado el Instrumento en tres pestañas diferenciadas. MATEMÁTICAS, ANÁLISIS ESPECTRAL y RESPUESTA EN FRECUENCIA. Se puede intuir fácilmente que se desarrollará en cada uno de ellos, pero aún se explica a continuación.

En la pestaña MATEMÁTICA, se pueden ver los valores de las medidas en amplitud que tienen las señales. Además, se han implementado operaciones básicas con señales.

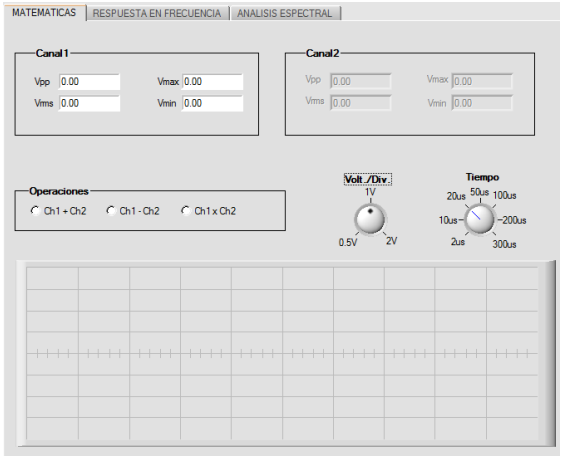


Figura 25: Instrumento virtual 2

En la pestaña ANÁLISIS ESPECTRAL, como se puede obviar, se muestra la señal espectral. Además, el espectro se podrá mostrar sin enventanar, o utilizando

diferentes tipos de enventanado como Hanning, Hamming, Triangular, Káiser y Blackmon.

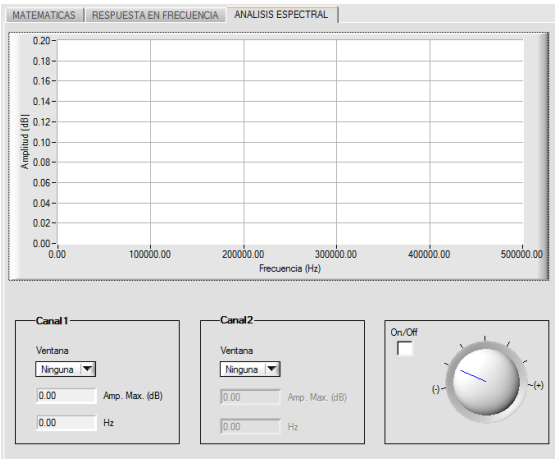


Figura 26: Instrumento virtual 3

Se pueden ver los valores de amplitud máxima de la señal (dB) y la frecuencia en el lóbulo principal.

Por último, y objeto principal de éste proyecto, está la pestaña RESPUESTA EN FRECUENCIA, que, como su propio nombre indica, calcula y muestra en tiempo real la respuesta en frecuencia del circuito.

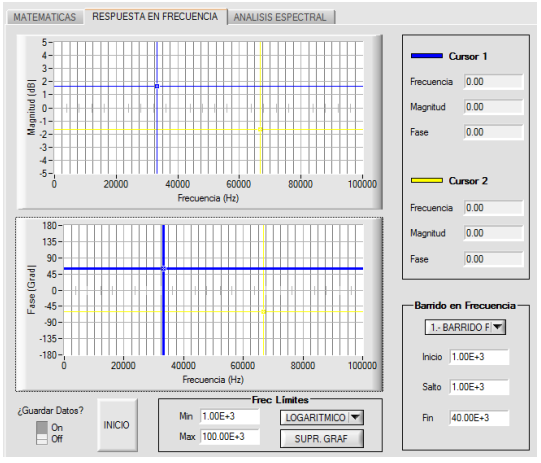


Figura 27: Instrumento virtual 4

En ella se pueden establecer los límites máximos y mínimos en los que se calculará la respuesta en frecuencia, haciendo un barrido en tiempo real de varias frecuencias equidistantes situadas entre el mínimo y el máximo establecido. Una vez calculados y mostrados los resultados, mediante los cursores se podrá hallar tanto la fase

como la magnitud, en las frecuencias deseadas, pudiendo así observar la frecuencia de corte, la caída en dB que se produce y la fase para cada una de las frecuencias.

5. RESULTADOS

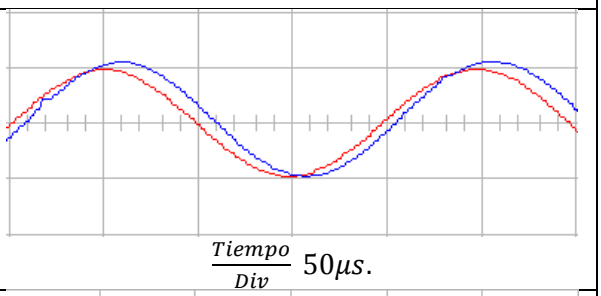
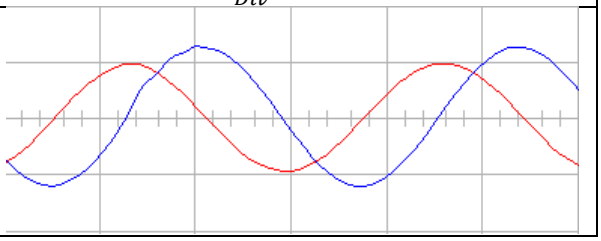
En este punto se analizarán los resultados obtenidos con el sistema diseñado en varios circuitos de prueba. Dichos circuitos de prueba serán un filtro paso-alto, y un filtro paso-bajo y un filtro paso-banda. Se utilizan las tres herramientas descritas en el apartado anterior para cada uno de los filtros.

5.1 RESULTADOS PESTAÑA DOMINIO DEL TIEMPO

Los filtros que se utilizaran para la prueba del instrumento son:

- Filtro 1: Un filtro paso-bajo en un Chevyshev, de orden 2, con frecuencia de corte de 20KHz.
- Filtro 2: Un filtro paso-alto es un Chebyshev, de orden 2, con frecuencia de corte de 40KHz.
- Filtro 3: Un filtro paso-banda Butterworth, de orden 2, con frecuencias de corte inferior y superior de 10 y 30KHz respectivamente.
-

Cabe destacar que los componentes utilizados para el montaje de los filtros no son los valores exactamente idénticos a los de las simulaciones teóricas, siendo valores aproximados, por lo que las frecuencias de corte podrán tener pequeñas variaciones con respecto a las fc teóricas. Ahora bien, se pasa al estudio de los filtros con el instrumento virtual.

	Filtro 1 (Paso Bajo)		
Frecuencia	Entrada filtro	Salida filtro	Imagen
5KHz	V_{pp}: 1.95 V_{rms}: 1.38	V_{pp}: 2.09 V_{rms}: 1.48	
15KHz	V_{pp}: 1.95 V_{rms}: 1.38	V_{pp}: 2.46 V_{rms}: 1.74	

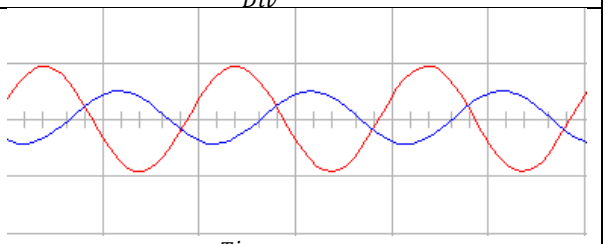
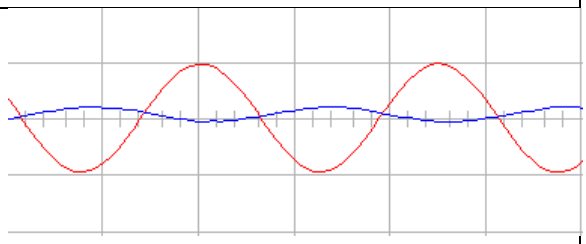
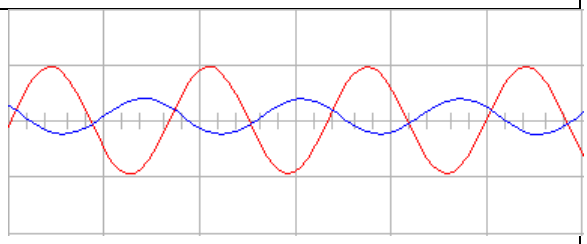
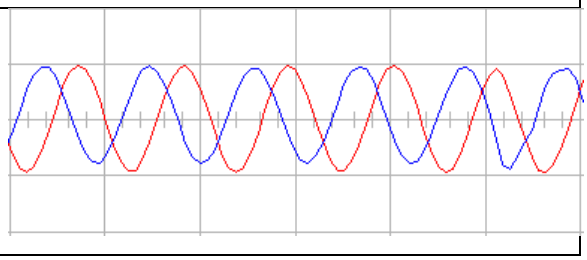
			$\frac{Tiempo}{Div}\ 20\mu s.$
25KHz	V_{pp}: 1.95 V_{rms}: 1.38	V_{pp}: 0.97 V_{rms}: 0.69	 $\frac{Tiempo}{Div}\ 20\mu s.$
	SEÑAL DE ENTRADA (CANAL 1)		
	SEÑAL DE SALIDA (CANAL 2)		

Tabla 4: Filtro paso bajo en etapa matemática

Se aprecia el efecto que tiene el filtro sobre la señal de entrada. En 5KHz, la amplitud de las señales es casi idéntica, mientras que, cuanto más se acerca a 20KHz, va aumentando, debido al rizado característico de los filtros Chevyshev. En 25KHz, la señal ya ha caído a la mitad de la tensión, por lo que hemos de suponer que el corte está más situado hacia los 25KHz que hacia los 20KHz.

	Filtro 2 (Paso Alto)		
Frecuencia	Entrada Filtro	Salida Filtro	Imagen
20KHz	V_{pp}: 1.97 V_{rms}: 1.39	V_{pp}: 0.3 V_{rms}: 0.21	 $\frac{Tiempo}{Div}\ 20\mu s.$
30KHz	V_{pp}: 1.96 V_{rms}: 1.39	V_{pp}: 0.69 V_{rms}: 0.49	 $\frac{Tiempo}{Div}\ 20\mu s.$
45KHz	V_{pp}: 1.99 V_{rms}: 1.41	V_{pp}: 1.79 V_{rms}: 1.26	

			$\frac{Tiempo}{Div} 20\mu s.$
	SEÑAL DE ENTRADA (CANAL 1)		
	SEÑAL DE ENTRADA (CANAL 2)		

Tabla 5: Filtro paso alto en etapa matemática

Al contrario que en el filtro paso-bajo, como cabe de esperar, en este filtro se aprecia la caída de tensión conforme decrece la frecuencia.

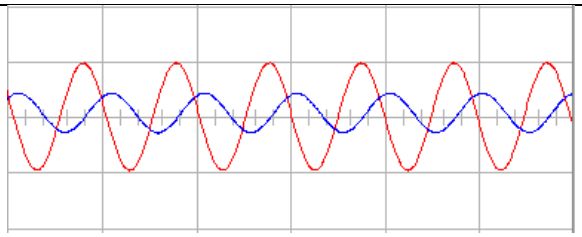
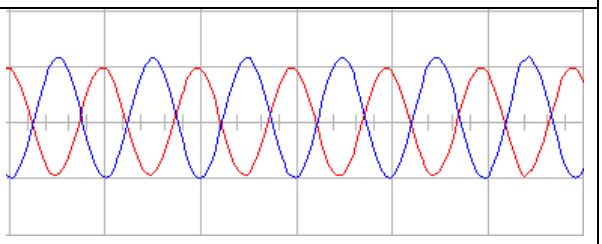
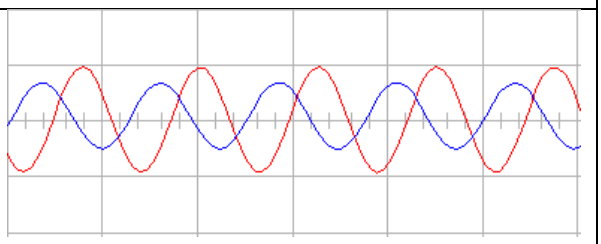
	Filtro 3 (Paso Banda)		
Frecuencia	Entrada Filtro	Salida Filtro	Imagen
5KHz	V_{pp}: 2.01 V_{rms}: 1.42	V_{pp}: 0.77 V_{rms}: 0.54	 $\frac{Tiempo}{Div}$ 200μs.
20KHz	V_{pp}: 1.95 V_{rms}: 1.38	V_{pp}: 2.17 V_{rms}: 1.53	 $\frac{Tiempo}{Div}$ 50μs.
40KHz	V_{pp}: 1.94 V_{rms}: 1.37	V_{pp}: 1.24 V_{rms}: 0.87	 $\frac{Tiempo}{Div}$ 20μs.
	SEÑAL DE ENTRADA (CANAL 1)		
	SEÑAL DE ENTRADA (CANAL 2)		

Tabla 6: Filtro paso-banda en etapa matemática

Se observa como la señal tiene su amplitud más alta entorno a 20KHz, que es justo el centro de la banda de paso de señal.

Con esta última prueba se da por finalizado el testeo de la etapa matemática, dando por satisfechas sus funciones. Se ha de notar que, aunque con esta etapa se puede intuir antes que tipo de filtro estamos, es una herramienta poco eficiente para caracterizar los circuitos. Para una mejor caracterización, pasaremos a comprobar las siguientes etapas.

5.2 RESULTADOS PESTAÑA ANÁLISIS ESPECTRAL

Para la prueba de esta etapa se utilizarán los mismos filtros usados en el punto anterior, y se comprobara su espectro para las mismas frecuencias.

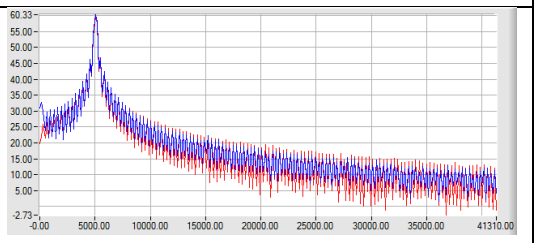
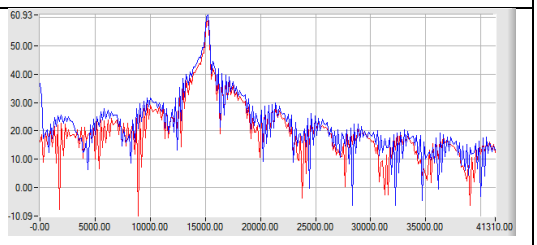
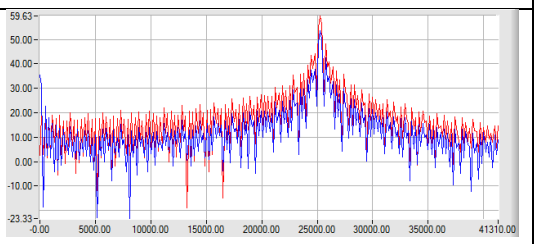
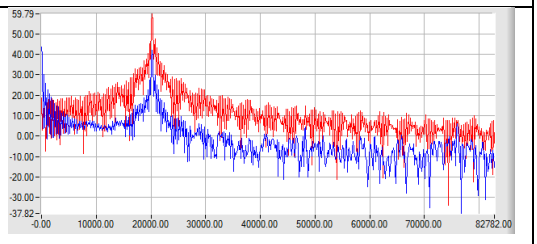
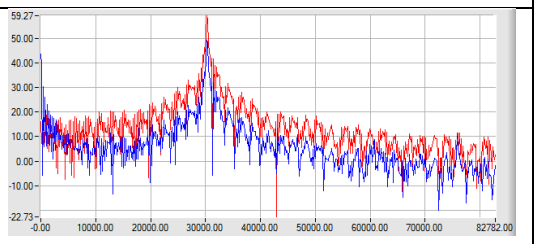
	Filtro 1 (Paso Bajo)		
Frecuencia	Entrada Filtro	Salida Filtro	Imagen
5KHz	$P: 60.34dB$	$P: 60.34dB$	
15KHz	$P: 60.38dB$	$P: 60.93dB$	
25KHz	$P: 59.63dB$	$P: 53dB$	
	SEÑAL DE ENTRADA (CANAL 1)		
	SEÑAL DE SALIDA (CANAL 2)		

Tabla 7: Filtro paso-bajo en etapa análisis espectral

Aquí, al igual que en la etapa temporal, se puede observar que se está ante un circuito paso bajo, pero no sabemos a ciencia cierta donde se sitúa la frecuencia de corte.

	Filtro 2 (Paso Alto)		
Frecuencia	Entrada Filtro	Salida Filtro	Imagen
20KHz	$P: 59.79dB$	$P: 41dB$	
30KHz	$P: 59.27dB$	$P: 50dB$	

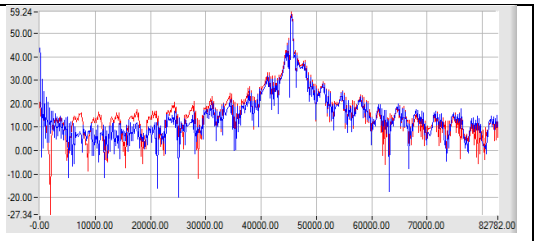
45KHz	$P: 59.24dB$	$P: 59.24dB$	
	SEÑAL DE ENTRADA (CANAL 1)		
	SEÑAL DE SALIDA (CANAL 2)		

Tabla 8: Filtro paso-alto en etapa análisis espectral

Claramente la potencia en la salida aumenta conforme aumenta la potencia, dejando claro que se trata de un filtro paso alto. Al igual que con el filtro anterior, en este tampoco podemos hallar la frecuencia de corte exactamente, por lo que se deberán realizar más pruebas al circuito con el instrumento virtual.

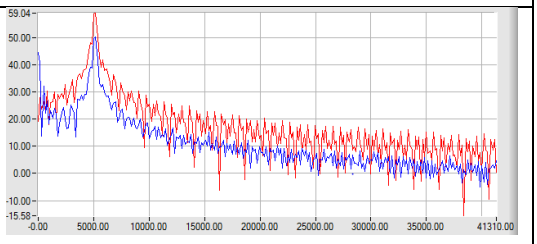
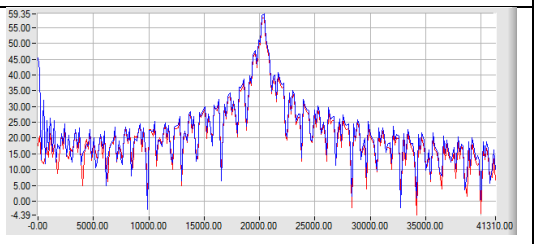
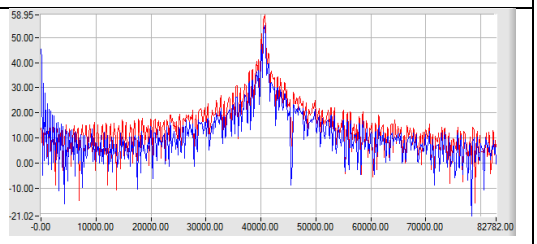
	Resultados Filtro		
Frecuencia	Entrada Filtro	Salida Filtro	Imagen
5KHz	$P: 59.04dB$	$P: 50dB$	
20KHz	$P: 59.35dB$	$P: 59.35dB$	
40KHz	$P: 58.95dB$	$P: 53dB$	
	SEÑAL DE ENTRADA (CANAL 1)		
	SEÑAL DE SALIDA (CANAL 2)		

Tabla 9: Filtro paso-banda en etapa análisis espectral

De nuevo, al igual que en la etapa temporal para este filtro, a la frecuencia de 20KHz se observa que la frecuencia de salida alcanza su máxima amplitud, igualando a la de entrada, por lo que se sabe que el centro de la banda está a 20KHz.

Pero, ¿se puede decir que están completamente caracterizados los tres circuitos? Es obvio que no, ya que al igual que en la etapa temporal, podemos suponer de qué tipo de filtros se tratan, pero todavía se sigue sin tener una caracterización completa. Para ello, se dispone de la más importante y última etapa, la etapa de función de transferencia.

5.3 RESULTADOS PESTAÑA FUNCIÓN DE TRANSFERNECIA

Este sea quizás la parte más importante de este proyecto, ya que es el objetivo principal del mismo. Ya se ha visto que mediante la etapa matemática y la etapa de análisis espectral, se pueden analizar circuito y, mediante los resultados, saber de qué tipo de circuito se trata. Pero, ¿por qué analizar tantos datos pudiendo analizar únicamente la respuesta en frecuencia del circuito? A continuación se verán los resultados de la respuesta en frecuencia con el instrumento virtual.

Hay varias formas de hallar la respuesta en frecuencia de un circuito entre las que se encuentran:

1) Analizar las señales temporales en sus máximos y mínimos, hallando su amplitud, y la diferencia entre máximos, para encontrar sus diferencias en amplitud y el desfase entre ellas.
2) Respuesta al impulso. Introducir un impulso a la entrada del circuito, lo suficientemente pequeño, y realizar un análisis espectral a la salida.
3) Algoritmo de Goertzel. Este algoritmo calcula la Discrete Fourier Transform (DFT) para valores concretos del índice espectral, en nuestro caso, para la frecuencia cuya respuesta queremos determinar. Ampliar información en [6].
4) Obtener todos los valores de la DFT, empleando el algoritmo Fast Fourier Transform (FFT). En este caso se determina la respuesta en frecuencia del circuito a la frecuencia donde hay mayor contenido espectral. Ampliar información en [6].

Este proyecto está realizado en las dos últimas formas. El tiempo del barrido es de aproximadamente 95s para ambos algoritmos.

Primero se realizó usando los máximos de la FFT, y posteriormente, se investigó una nueva forma de realizarlo mediante el algoritmo de Goertzel. El análisis de la señal

temporal ha sido descartado debido a que la pureza de la señal interfería para el desarrollo de esta, provocando picos de amplitud no deseados que afectan a la hora de hallar los máximos o mínimos de la señal. En cuanto a la respuesta impulsional, se ha dejado como opción a desarrollar en un futuro.

Se analizará, para seguir el mismo orden que se llevaba hasta ahora, el filtro paso bajo. Para realizar dicho análisis se hará un barrido en frecuencia entre 1KHz y 40KHz. Primero se comenzará realizando un barrido he implementado la FFT, para luego continuar realizando un barrido he implementar el algoritmo de Goertzel.

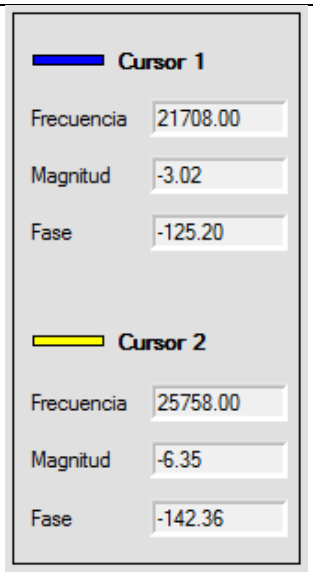
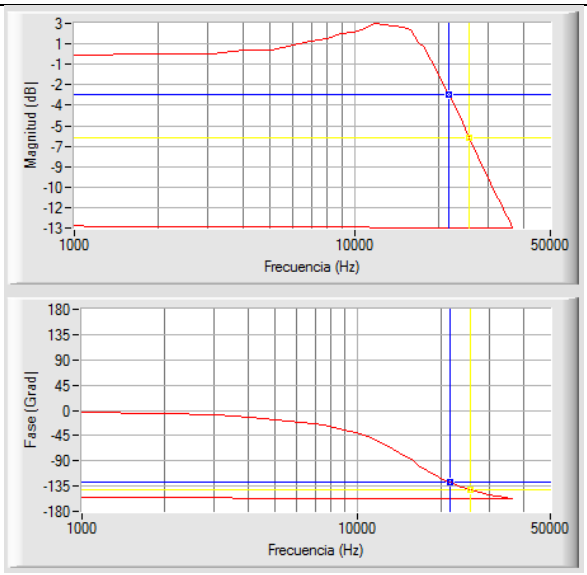
	Filtro 1 (Paso-bajo)	
Frecuencia	Markers	Imagen
1KHz → 40KHz		

Tabla 10: Respuesta en frecuencia filtro Paso Bajo (barrido + FFT)

Con esta herramienta ya si se puede deducir muy fácilmente que se está ante un filtro paso bajo.

La frecuencia de corte, que es la frecuencia a la que la señal cae al doble de la potencia, o -3dB, se sitúa en 21,7KHz. Quiere decir que hay algo de variación en la frecuencia de corte con respecto al filtro teórico, pero era de esperar debido a que lo componentes para el montaje del filtro no corresponden exactamente con los calculados.

Al ser un filtro de orden 2, la transición entre la banda de paso y la banda de stop es muy lenta. Si se aumentase el orden del filtro, se conseguiría una banda de transición más rápida y con ello un filtro más preciso.

Respecto al desfase, se observa como aumenta conforme se aumenta la frecuencia.

A continuación, se analizara la respuesta en frecuencia del filtro paso alto. Para ello, realizaremos un barrido en frecuencia entre 20KHz y 80KHz.

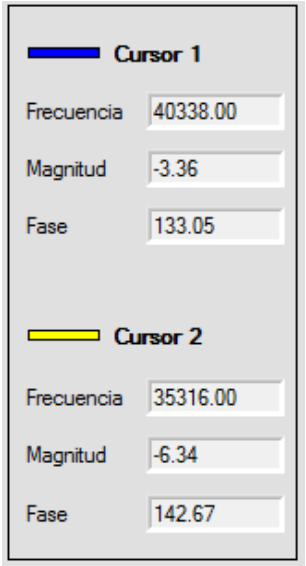
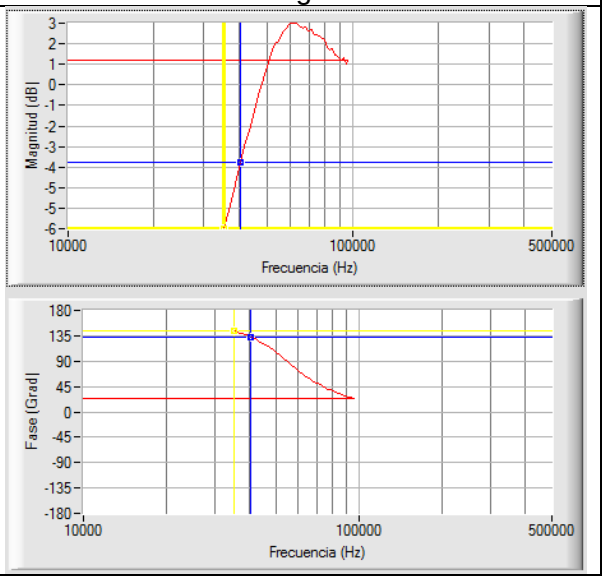
	Filtro 2 (Paso-alto)	
Frecuencia	Markers	Imagen
20KHz → 80KHz		

Tabla 11: Respuesta en frecuencia filtro Paso Alto (barrido + FFT)

Se observa como la frecuencia de corte del filtro paso bajo si se asemeja mucho más al filtro teórico, estando está en 40,3KHz. Al igual que en el filtro paso bajo la caída en frecuencia es bastante lenta, debido al orden del filtro.

En este filtro, el desfase se encuentra en las frecuencias más bajas, aproximándose a cero para las frecuencias más altas.

Para finalizar el apartado de resultados, será probado un circuito paso banda, con unas frecuencias de corte teóricas de 10 a 30KHz. El barrido en frecuencia se realizara entre 1 y 50KHz.

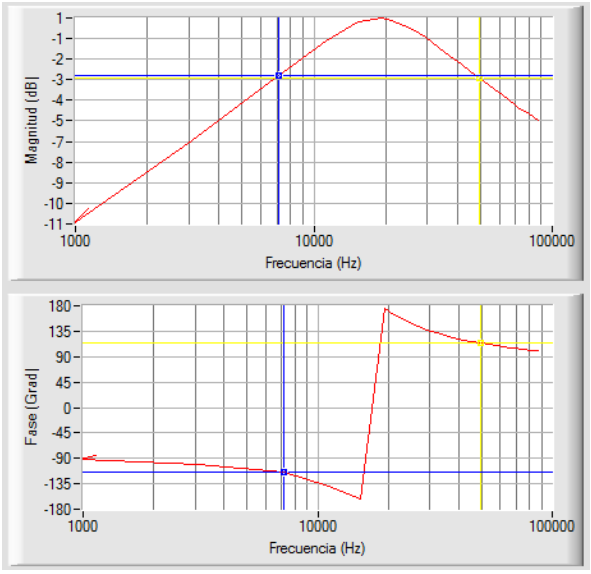
	Resultado filtro paso-banda (10-30KHz)	
Frecuencia	Markers	Imagen
1KHZ → 100KHz	Cursor 1 Frecuencia 7128.00 Magnitud -2.79 Fase -115.30 Cursor 2 Frecuencia 49572.00 Magnitud -2.97 Fase 111.18	

Tabla 12: Respuesta en frecuencia filtro Paso Banda (barrido + FFT)

La frecuencia de corte inferior está muy próxima a la frecuencia de corte teórica. En cambio, la frecuencia de corte superior si está más por encima de la frecuencia de corte teórica y, al igual que con los filtros anteriores, puede ser debido a la inexactitud de los componentes. El filtro alcanza su máxima amplitud a los 19KHz, que está muy próximo a los resultados que se mostraban en las etapas anteriores.

En cuanto a la fase, para las frecuencias bajas se tiene un desfase negativo, mientras que para las frecuencias bajas, el desfase es positivo.

A continuación se mostrarán los resultados utilizando el algoritmo de Goertzel.

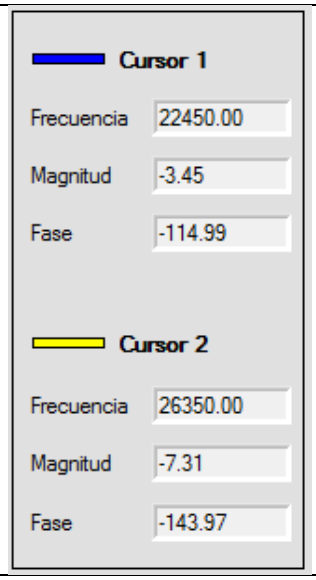
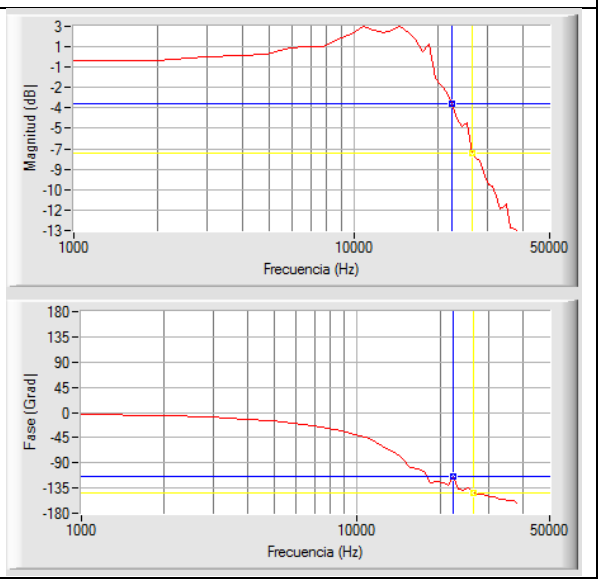
	Filtro 1 (Paso Bajo)	
Frecuencia	Markers	Imagen
1KHz → 40KHz		

Tabla 13: Respuesta en frecuencia filtro Paso Bajo (barrido + alg.Goertzel)

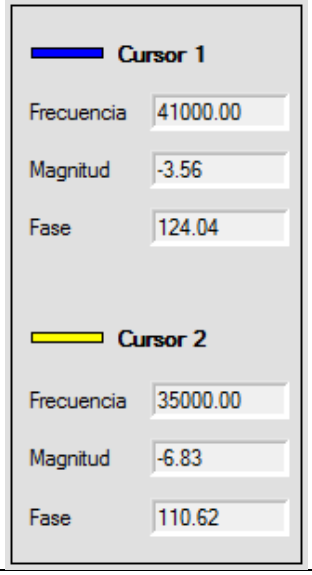
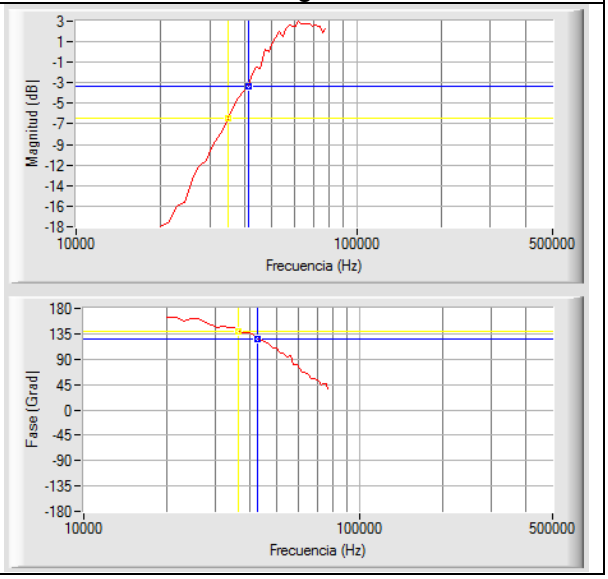
	Filtro 1 (Paso Alto)	
Frecuencia	Markers	Imagen
20KHz → 80KHz		

Tabla 14: Respuesta en frecuencia filtro Paso Alto (barrido + alg.Goertzel)

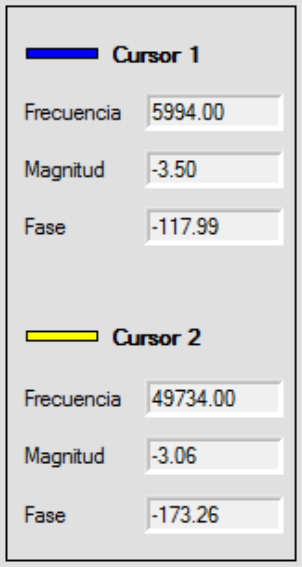
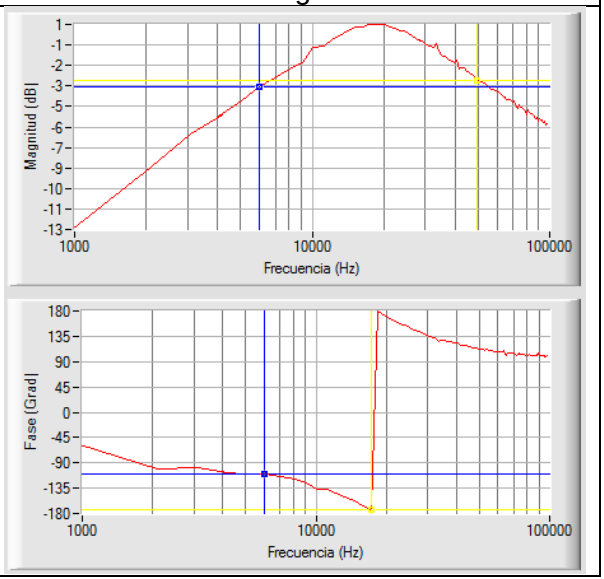
	Filtro 1 (Paso-banda)	
Frecuencia	Markers	Imagen
1 → 50KHz		

Tabla 15: Respuesta en frecuencia filtro Paso Banda (barrido + alg.Goertzel)

Se puede observar como los resultados entre el algoritmo de Goertzel y la FFT son similares. Los picos en las gráficas son debidos a pequeñas variaciones en el espectro. También se ha de tener en cuenta que los datos que se usan para cada uno de las dos implementaciones no son los mismos. La diferencia radica en la complejidad de cálculo que requieren una y otra. Estas complejidades vienen dadas en la siguiente tabla:

Complejidad alg. Goertzel vs FFT:	
Alg. Goertzel	N^2
FFT	$N \cdot log_2(N)$

Tabla 16: Complejidad alg. Goertzel vs FFT. *N es el número de puntos donde se realiza el análisis espectral*

Según los datos de la tabla anterior, la complejidad de la FFT es menor que la del algoritmo de Goertzel pero, la ventaja de este último, es que solo se ha de calcular para un punto N específico, por lo que su complejidad se reduce significativamente. Por tanto, para un análisis de pocos puntos, el algoritmo de Goertzel es más eficiente que la FFT.

Para verificar que estos datos son correctos, se ha realizado un análisis externo de los datos, utilizando Matlab. Dichos resultados se pueden comprobar en el ANEXO IV: ANALISIS CON MATLAB DE LOS RESULTADOS OBTENIDOS.

5.4 CARACTERÍSTICAS DEL INSTRUMENTO VIRTUAL

Se resumen a continuación las principales características del instrumento virtual desarrollado:

Canales Salida	1 Canales
Canales de Entrada	2 Canales
Ancho de banda de medida de la respuesta en frecuencia	1 KHz -140KHz
Frecuencia de muestreo	666 Kmuestras/s
Resolución Vertical	12 bits
Rango de amplitud	0V - 2V
Tiempo de barrido máximo:	278 segundos (realizando barrido de todo el ancho de banda de medida con resolución de 1KHz)
Operaciones Matemáticas adicionales	FFT, adición, multiplicación, diferencia

Tabla 17: Características finales del instrumento

6. CONCLUSIONES Y LÍNEAS FUTURAS

6.1 CONCLUSIONES

1.- Se ha cumplido el objetivo principal de este proyecto, que era la de crear un instrumento de un coste relativamente bajo, que pueda ayudar a los alumnos a dar un punto de vista más practico en el análisis de circuitos lineales.

2.- Las principales limitaciones del instrumento desarrollado son el ancho de banda de medida del instrumento virtual y el tiempo de barrido máximo.

3.- Es un instrumento fácil de modificar y reconfigurar para hacer otro tipo de medidas.

4.- Se ha podido observar cómo se obtienen la respuesta en frecuencia de varios circuitos (Paso Alto, Paso Banda y Paso Bajo).

6.2 LÍNEAS FUTURAS

Hay varias opciones que se pueden tomar en el futuro para la mejora de este prototipo instrumento virtual.

1.- Se ha visto que una de los principales inconvenientes que se han encontrado en este proyecto, ha sido el de obtener un margen de frecuencias más amplio. En próximas versiones, para paliar esta desventaja, se pueden usar conversores A/D

externos e incluso memorias externas, controladas con el propio Arduino. Aún a riesgo de encarecer el coste del proyecto, es posible que repercuta en una amplia mejora del espectro de medida del dispositivo.

2.- Aumento de las tensiones de entrada al dispositivo. Ya que en un principio se consideró la posibilidad de realizar un divisor de tensión y corregir mediante código, se descartó la opción debido a que con señales de menor amplitud se perdería mucha resolución. En el futuro podrán emplearse potenciómetros electrónicos u otro método para el control de las amplitudes de entrada.

3.- Ampliar las funciones que pueda ejercer el dispositivo, ampliando el número de herramientas de este.

4.- Otro punto a tener en cuenta en el futuro, sería establecer una conexión a internet vía WIFI o bien vía Ethernet, con la que poder manejar el dispositivo a distancia, o enviar los datos telemáticamente.

5.- Obtener la respuesta en frecuencia mediante la respuesta al impulso, generando un pulso lo más estrecho posible, que posteriormente permita analizar su espectro.

6.- Mejorar la adaptación de señal y aumentar el rechazo al modo común usando en la etapa de adquisición amplificadores de instrumentación, mejorando la inmunidad.

7. BIBLIOGRAFÍA

1. Goldberg, Harold. *What is Virtual Instrumentation?* IEEE Instrumentation & Measurement Magazine. 2010, pp. 10-13.

2. Arduino Due. ©2016 [consulta: 5 de Febrero de 2016.] Disponible en: https://www.arduino.cc/en/Main/ArduinoBoardDue.

3. *What is Arduino? ©2016 [consulta: 5 de Febrero de 2016.] Disponible en:* https://www.arduino.cc/en/Guide/Introduction.

4. *Wiring.* ©2011 [consulta: 5 de Febrero de 2016.] Disponible en: http://wiring.org.co/.

5. Proccesing. ©2004 [consulta: 5 de Febrero de 2016.] Disponible en: https://processing.org/.

6. Oppenheim, Alan V., Schafer, Ronald W. y John, Buck R. *Discrete-Time Signal Processing*. 2nd ed. New Jersey: Prentice Hall, 1998. ISBN 0-13-754920-2

7. Goertzel, G. *An Algorithm for the Evaluation of Finite Trigonometric Series*. Vol. 65 Monthly: American Math, 1958.

8. *Instrumentación Virtual e Instrumentación Tradicional. ©2016 [consulta: 15 de Julio de 2015.] Disponible en:* http://www.ni.com/white-paper/4757/es/.

9. Analog Device. *CMOS, 125MHz Complete DDS Synthesizer AD9850.* 2004.

10. Arduino. *shiftOut.* ©2016 [consulta: 10 de Agostode 2015.] Disponible en: https://www.arduino.cc/en/Reference/ShiftOut.

11. *SAM3X-Arduino Pin Mapping. ©2016 [consulta: 10 de Febrero de 2016.] Disponible en:* https://www.arduino.cc/en/Hacking/PinMappingSAM3X.

12. Atmel. *SAM3X/SAM3A Series.* 2015.

13.*¿Que es LabWindows/CVI?* ©2016 [consulta: 10 de Julio de 2015.] Disponible en: http://www.ni.com/lwcvi/whatis/esa/.

14. *Multithreading in LabWindows™/CVI. ©2016 [consulta: 10 de Febrero de 2016.] Disponible en:* http://www.ni.com/white-paper/3663/en/.

ANEXO I: MANUAL DE USUARIO

Esto es un manual de usuario con la explicación de cada uno de los controles para el manejo de este instrumento virtual.

Antes de comenzar a usar el instrumento, es necesario realizar bien las conexiones previas. Todas las masas del aparato están conectadas, por lo que no será necesario que las puntas utilicen la conexión de masa. Asegurarse también de que todas las polarizaciones estén bien realizadas.

Para la realización de mediciones la fuente de alimentación siempre ha de estar conectada, o usar una fuente de alimentación simétrica externa.

Para la conexión con el ordenador, usaremos un cable miniUSB. La siguiente figura muestra las conexiones externas del aparato.

Figura 28: Manual de usuario. Conexiones delanteras

1) MiniUSB controlador SAM3X Arduino **(no utilizar)**
2) MiniUSB controlador Atmega Arduino. Este miniUSB es el usado para la conexión con el PC.
3) Alimentación Arduino DUE **(no utilizar).** Arduino se alimenta a través del puerto miniUSB.
4) Conector de salida del generador de onda.
5) Conector de entrada Canal 1.
6) Conector de entrada Canal 2.

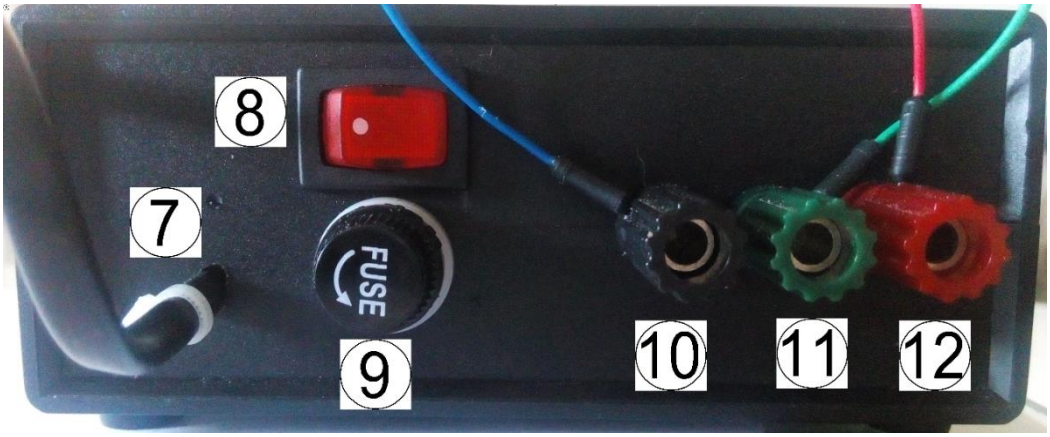


Figura 29: Manual de usuario. Conexiones traseras

7) Cable de alimentación 220VAC.
8) Interruptor de encendido.
9) Fusible.
10) Alimentación -12V DC.
11) Toma de tierra.
12) Alimentación +12V DC.

Todas las masas del circuito están unidas interiormente, por lo que no será necesario la utilización del cable de masa de la sonda.

¡Atención!: Comprobar muy bien las polarizaciones antes del encendido del aparato para evitar posibles cortocircuitos.

A continuación se detallan las partes para el manejo correcto del software.

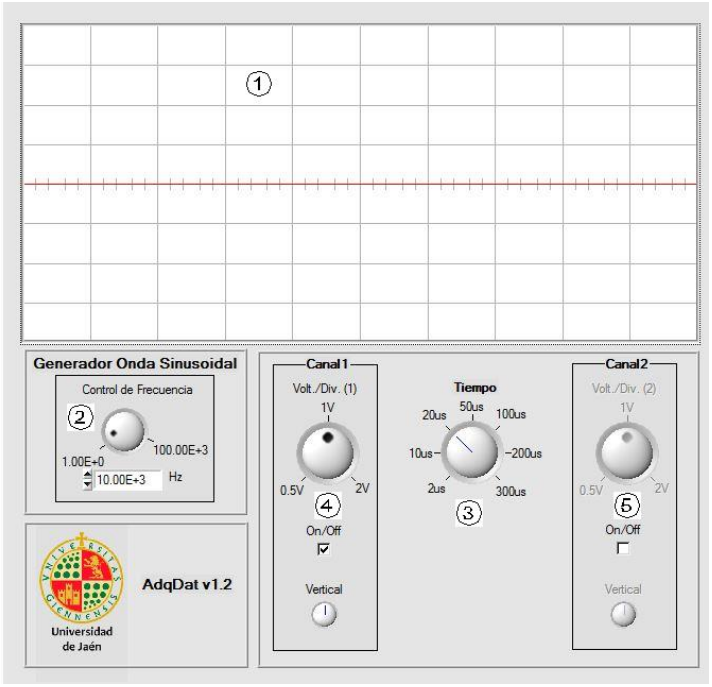


Figura 30: Manual de usuario 1

1) Gráfica de la señal temporal.
2) Control del generador de frecuencia.
3) Control eje de tiempo de la gráfica de señal temporal.
4) Control eje vertical del canal 1 y activación/desactivación del mismo.
5) Control eje vertical del canal 2 y activación/desactivación del mismo.

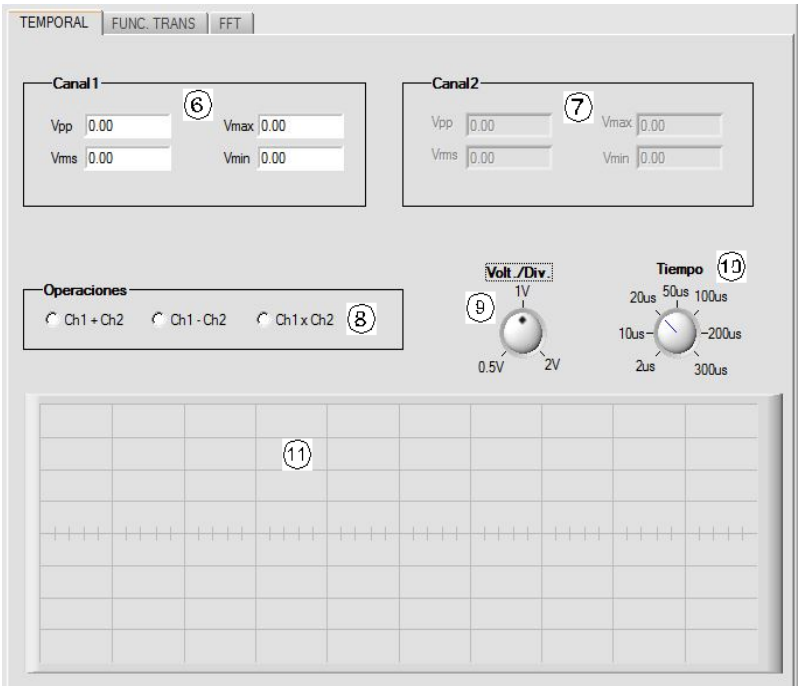


Figura 31: Manual de usuario 2

6) Medidas de tensión canal 1.
7) Medidas de tensión canal 2.
8) Operaciones entre señales. Suma resta y multiplicación de señales.
9) Control eje vertical de la gráfica de operación de señales.
10) Control del eje horizontal de la gráfica de operación de señales.
11) Grafica de operación de señales. En ella se muestran las operaciones que se realizan.

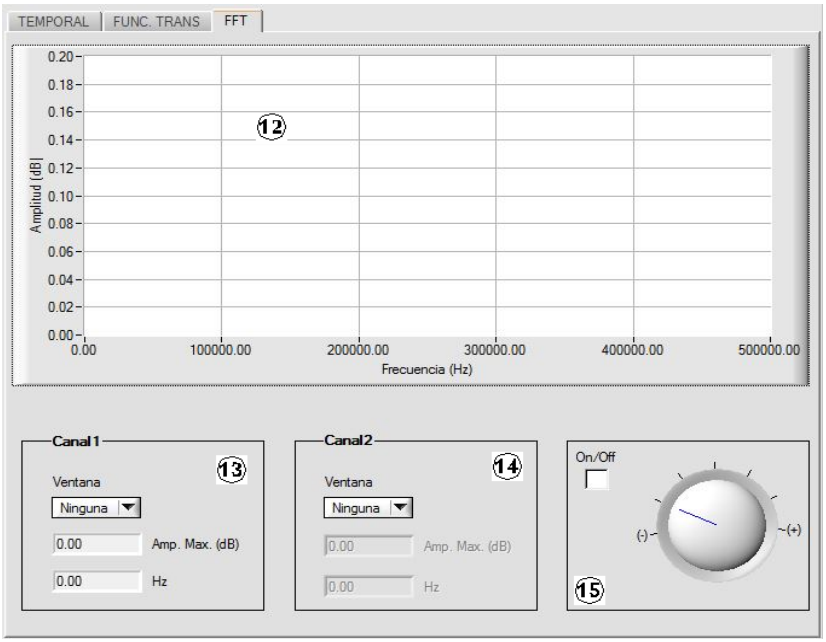


Figura 32: Manual de usuario 3

12) Gráfica para representación de la FFT.

13) Tipos de enventanado para realización de FFT en el canal 1, nivel de amplitud en decibelios y muestra de la frecuencia con mayor potencia.

14) Tipos de enventanado para realización de FFT en el canal 2, nivel de amplitud en decibelios y muestra de la frecuencia con mayor potencia.

15) Activación y desactivación para la realización de la FFT y modificación del eje horizontal de la gráfica de FFT.

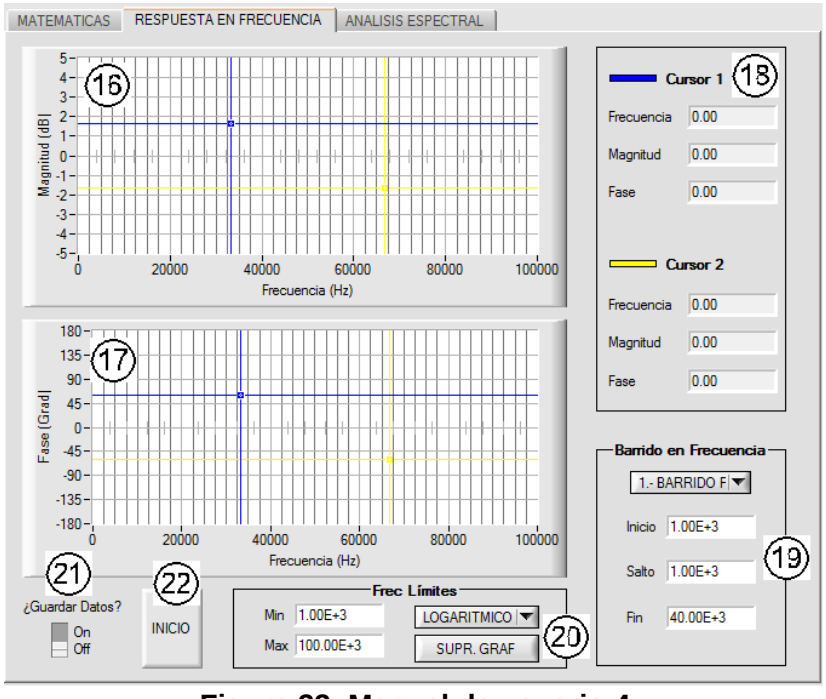


Figura 33: Manual de usuario 4

16) Gráfica de magnitud de la respuesta en frecuencia.
17) Gráfica de fase de la respuesta en frecuencia.
18) Medidas de los cursores 1 y 2 de las gráficas de magnitud y fase.
19) Introducción del límite mínimo y máximo para el cálculo de la respuesta en frecuencia y selección del tipo de cálculo para hallar esta.
20) Introducción del límite mínimo y máximo para la configuración de las gráficas de la respuesta en frecuencia. Configuración del tipo de gráfica y limpieza de estas.
21) Opción para guardar los datos de la señal temporal para posteriormente poder analizarlos externamente. Para su uso es necesario realizar el análisis con el algoritmo de Goertzel seleccionado,
22) Botón para iniciar el barrido de frecuencias. Antes de pulsar este botón, se ha de asegurar que todos los parámetros estén bien establecidos, y que el generador de frecuencia este a la frecuencia inicial del barrido.

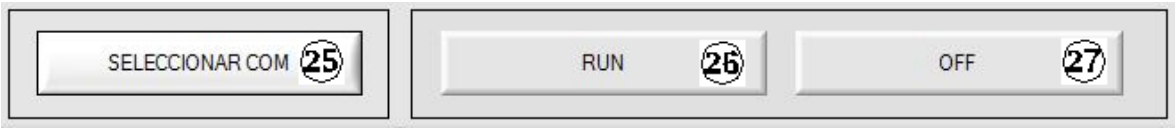


Figura 34: Manual de usuario 5

25) Botón para seleccionar el puerto USB. Hay que seleccionar el puerto antes de ejecutar el programa.
26) Botón para el inicio o pausa del programa.
27) Botón para cerrar el programa.

Figura 35: Manual de usuario 6

28) Al seleccionar el botón para cambiar el COM, aparecerá este cuadro para poder seleccionar el COM correspondiente.

En caso de no seleccionar el COM correcto, aparecerá un error. Por favor, compruebe de nuevo en que puerto COM está conectado el hardware y ejecute de nuevo el programa.

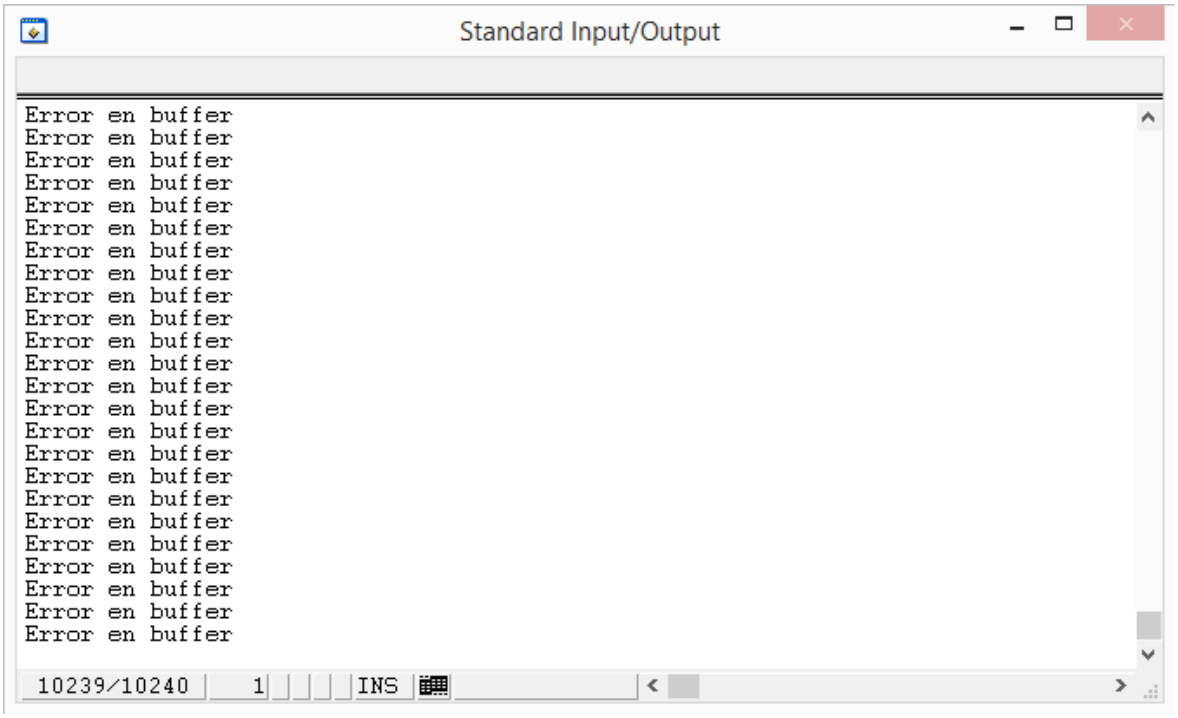


Figura 36: Manual de usuario 7

ANEXO II: MANUAL DEL PROGRAMADOR

A continuación se detallarán las funciones más importantes del programa.

```
/*************************************************************/
/*                                                           */
/*Función selecport_ok                                       */
/*                                                           */
/*Utilizada para la configuración del puerto serie           */
/*                                                           */
/*************************************************************/
int CVICALLBACK selecport_ok (int panel, int control, int event,
            void *callbackData, int eventData1, int eventData2)
```

Tabla 18: Manual del programador (Selección de puerto)

Con selecport, se configura el puerto COM, como el número de puerto, la velocidad en baudios, número de paridad, etc. Todo esto está vinculado al segundo panel del programa, que se activa cuando queremos seleccionar el COM.

```
/*************************************************************/
/*                                                           */
/*Funcion lecDat                                             */
/*                                                           */
/*Variables: conError, functionId2, strLen, buffer[strLen]   */
/*                                                           */
/*Lectura de los datos mediante puerto serie, escritura de datos  */
/*en vectores correspondientes y llamada a hilo secundario para   */
/*comenzar el procesado de datos.                            */
/*                                                           */
/*************************************************************/
 int CVICALLBACK lecDat (void *functionData)
```

Tabla 19: Manual del programador (lectura de datos)

Con lecDat se leen los datos desde el puerto serial. La lectura del buffer tiene programado un mensaje de error en caso de que la lectura de este no sea correcta. Se asignan los datos a sus correspondientes vectores. Se aplica un filtro paso bajo para la mejora de la señal de entrada.

```
/*************************************************************/
/*                                                           */
/*Funcion procDat                                             */
/*                                                           */
/*Variables: f, constants                                    */
/*                                                           */
/*Procesado de los datos para obtener maximos y minimos en los    */
vectores, cálculo de la transformada de Fourier y detección de la */
frecuencia                                                     */
/*                                                           */
/*************************************************************/
int CVICALLBACK procDat (int reserved, int theTimerId, int event, void
*callbackData, int eventData1, int eventData2)
```

Tabla 20: Manual del programador (procesado de datos)

En procDat, se calcula el máximo de los vectores y se aplica la FFT con los distintos enventanados. En esta función también se incluye el cálculo de suma resta y multiplicación de señales.

```
/*****************************************************************/
/*                                                               */
/*Funcion functrans                                              */
/*                                                               */
/*Variables: CALC, intv, tam, contar, dif, frec, b[], info[].    */
/*                                                               */
/*Calculo de la función de transferencia mediante un barrido de  */
frecuencias+FFT.                                                 */
/*                                                               */
/***************************************************************** */
int CVICALLBACK functrans (void *functionData)
```

Tabla 21: Manual del programador (respuesta en frecuencia, barrido + FFT)

```
/*****************************************************************/
/*                                                               */
/*Funcion functrans1                                             */
/*                                                               */
/*Variables: CALC, gDat, intv, contar, k, stat, stat1, fileHdl,  */
posFile, dif, frec, coef, w, q00, q1, q02, q10, q11, q12, real0, */
real1, imag0, imag1, mag0, mag1, *ag0, *ag1, *vGDatEnt, *vGDatSal,*/
b[], info[], ficheroDatosEnt[], ficheroDatosSal[600]             */
/*                                                               */
/*Calculo de la función de transferencia mediante un barrido de  */
/*frecuencias + algoritmo de Goertzel. Tambien se implementa el  */
/*guardado de datos en fichero.                                  */
/*                                                               */
/*****************************************************************/
int CVICALLBACK functrans1 (void *functionData)
```

Tabla 22: Manual del programador (respuesta en frecuencia, barrido + algoritmo de Goertzel)

Con functrans y functrans1 se calcula la respuesta en frecuencia usando un barrido con FFT y un barrido con el algoritmo de Goertzel, respectivamente.

ANEXO III: ESQUEMÁTICO DE CIRCUITOS

A continuación se presentan los esquemáticos de los circuitos utilizados en este proyecto, para su posterior montaje en placas perforadas o impresas. En el caso de este proyecto, se han implementado en placas perforadas.

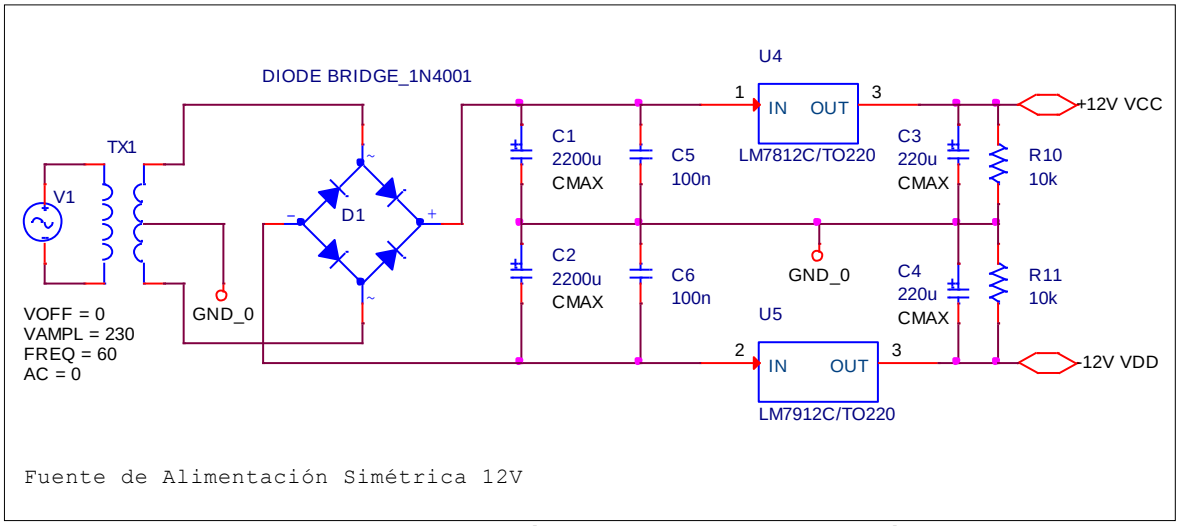


Figura 37: Esquemático fuente de alimentación

En la figura 36 se muestra la fuente de alimentación simétrica, de $\pm 12Vcc$, y una corriente máxima de salida de 0.5A.

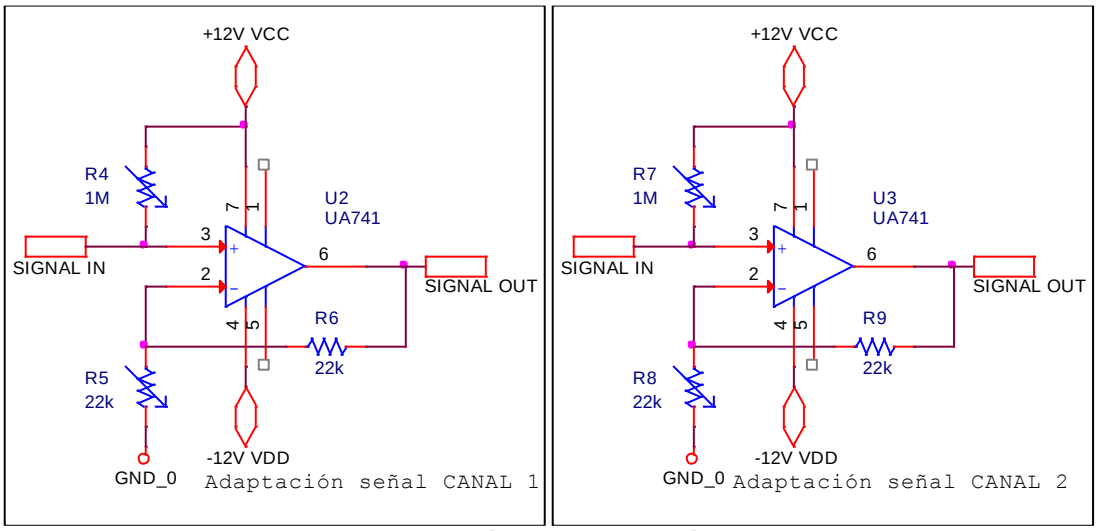


Figura 38: Esquemático Adaptación Canal 1 y 2

Se puede observar que la adaptación se hace de la misma manera, como es lógico, en cada canal. Las resistencias R4 y R7 son usadas para cambiar el punto medio de tensión de la señal, mientras que R5 y R8 son usadas para ajustar la amplitud de la señal.

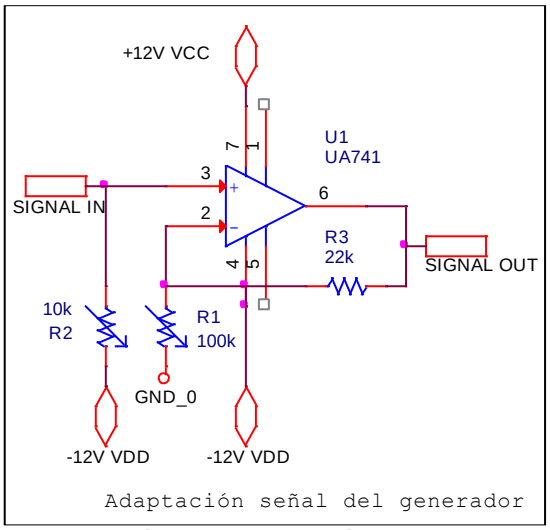


Figura 39: Esquemático Adaptación Salida del Generador

Dado que el sintetizador solo genera valores positivo de tensión, se ha de adaptar la señal para que el punto medio de amplitud sea cero, para lo que se utiliza R2, mientras que R1 es utilizada para modificar la ganancia.

ANEXO IV: ANALISIS CON MATLAB DE LOS DATOS OBTENIDOS

Para la realización de este anexo, se han capturado los datos de las señales temporales de entrada y salida, para su posterior análisis con Matlab.

Se ha probado realizando un barrido sobre el filtro paso-bajo visto en anteriores partes de este proyecto, mostrando los siguientes resultados.

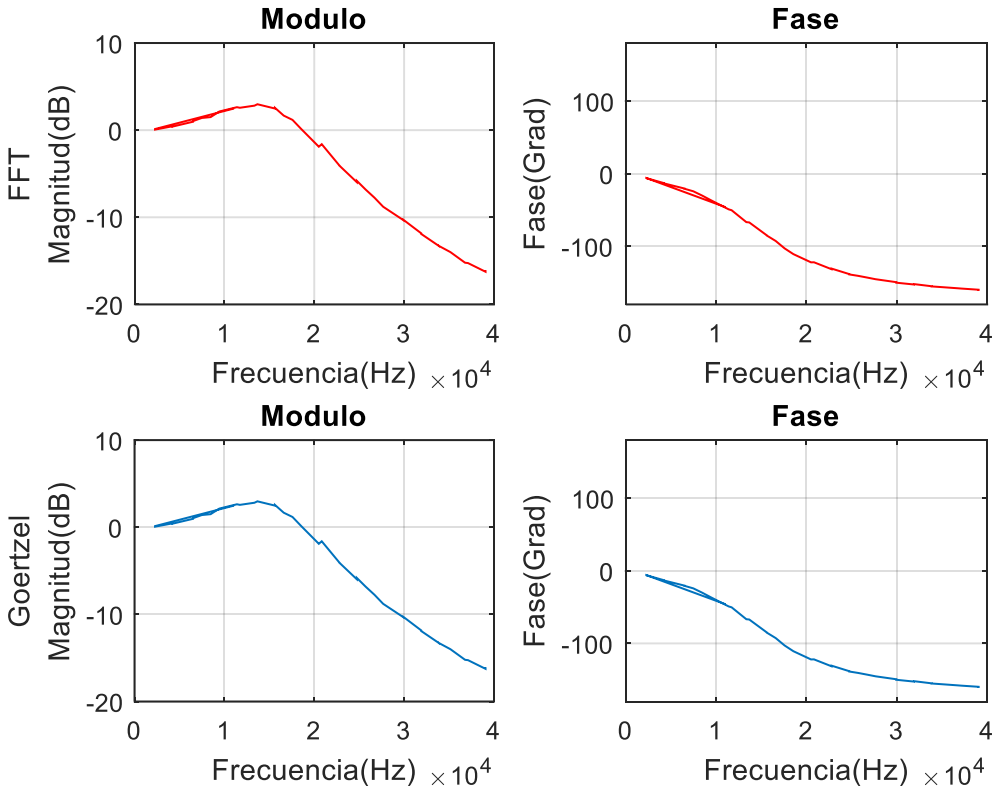


Figura 40: Algoritmo de Goertzel vs FFT (MATLAB)

En la parte superior, en rojo, se presenta la función de transferencia usando barrido de frecuencias + FFT, mientras que abajo se ha usado el algoritmo de Goertzel.

ANEXO V: PRESUPUESTO

Producto	Cantidad	Descripción	Unidades	Precio / unidad	Precio
1.0 FUENTE DE ALIMENTACIÓN SIMETRICA					
FA0001	1	Transformador 12 0 12		10,21 €	10,21 €
FA0002	5	Terminales Conex. Hembra		0,06 €	0,30 €
FA0003	6	Diodos 1N4001		0,05 €	0,30 €
FA0004	2	Cond. Electr. 2200uF 50V		1,32 €	2,64 €
FA0005	2	Resistencias 10KOhm		0,05 €	0,10 €
FA0006	1	Regulador LM7812		0,19 €	0,19 €
FA0007	1	Regulador LM7912		0,19 €	0,19 €
FA0008	2	Disipador		0,63 €	1,26 €
FA0009	2	Cond. Electr. 220uF 25V		0,48 €	0,96 €
FA0010	1	Interruptor		0,60 €	0,60 €
FA0011	1	Portafusible		0,63 €	0,63 €
FA0012	1	Fusible		0,13 €	0,13 €
FA0013	1	Cable		1,00 €	1,00 €
FA0014	3	Conec. Banana Hembr.		0,50 €	1,50 €
2.0 ADAPTACIÓN DE SEÑAL					
AS0001	6	Potenciometros		0,69 €	4,14 €
AS0002	1	Res. 22KOhm		0,05 €	0,05 €
AS0003	2	Res. 10KOhm		0,05 €	0,10 €
AS0004	3	Amp. Op. UA741		0,15 €	0,45 €
AS0005	3	Conectores BNC		0,60 €	1,80 €
3.0 MICROELECTRÓNICA					
ME0001	1	Arduino DUE R3		8,63 €	8,63 €
ME0002	1	Sint. DDS AD9850		9,00 €	9,00 €
4.0 OTROS					
OT0001	1	Placa ProtShield Aduino		1,80 €	1,80 €
OT0002	2	Placa Perforada		0,80 €	1,60 €
				Subtotal	45,98 €
El precio podrá variar según condiciones de mercadado			**21,00%**	IVA	9,66 €
				Costes de Envio	
				Seguro	
				Total	**55,64 €**

ANEXO VI: ÍNDICE DE FIGURAS

ANEXO VII: ÍNDICE DE TABLAS