Trabajo Fin de Grado
Grado de Ingeniería de las Tecnologías Industriales

Clasificador de Impedancias borroso utilizando la plataforma Arduino

Autor: Carlos Bordons Martínez

Tutor: José María Maestre Torreblanca y Juan Manuel Escaño González

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Trabajo Fin de Grado
Grado de Ingeniería de las Tecnologías Industriales

Clasificador de Impedancias borroso utilizando la plataforma Arduino

Autor:
Carlos Bordons Martínez

Tutor:
José María Maestre Torreblanca y Juan Manuel Escaño González

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Trabajo Fin de Grado: Clasificador de Impedancias borroso utilizando la plataforma Arduino

Autor: Carlos Bordons Martínez
Tutor: José María Maestre Torreblanca y Juan Manuel Escaño González

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

A mi familia, amigos y tutores que me han ayudado y apoyado mucho durante toda mi carrera y en particular durante este TFG

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Objetivo

El objetivo de este proyecto es obtener un sensor de impedancia de bajo coste y tamaño reducido basándose en la plataforma Arduino y utilizarlo para clasificar distintas impedancias mediante lógica borrosa (Fuzzy Logic), se pretende utilizar este método para clasificación biométrica y su aplicación en el campo de la domótica, aunque se trata de un objetivo a largo plazo y se llegará hasta el apartado de clasificación de la bioimpedancia.

1.2. Resumen

En este documento se exponen las principales herramientas, demostraciones y base teórica necesarias para alcanzar nuestro objetivo, así como los resultados obtenidos. Se tratan los temas de la transformada de Fourier para trabajar en el dominio de la frecuencia y de la impedancia eléctrica de un circuito y su utilidad para diferenciar distintos elementos, además se realiza una introducción a las plataformas Arduino y Matlab para la adquisición y posterior tratamiento de los datos.

Se expondrán las propiedades de la transformada de Fourier y su aplicación al tratamiento de los datos en el capítulo 2, la tecnología escogida para la adquisición de datos es la plataforma Arduino, debido a su bajo coste, que se explica con mayor detalle en el capítulo 3.

El capítulo 4 versa sobre la impedancia y como medirla correctamente utilizando distintos métodos y en el capítulo 5 se explica la obtención de el diagrama de Bode de una impedancia, que será el método utilizado para diferenciar una impedancia de otra y el método borroso utilizado para clasificar los diferentes diagramas de Bode de las impedancias medidas.

Para finalizar en el capítulo 6 se muestra el software Matlab utilizado para el tratamiento de datos, y en los capítulos 7 y 8 se muestran los expe-

rimentos realizados y los resultados obtenidos.

1.3. Antecedentes

La identificación e interacción biométrica es un tema en continuo desarrollo debido a la revolución de las pantallas táctiles y el gran campo de investigación que ofrece, si nos centramos en la clasificación mediante impedancia y en particular con sistemas de bajo coste como Arduino, se encuentran entre otros el proyecto *Touché for Arduino* [1], y el documento *Touché: Enhancing Touch Interaction on Humans, Screens, Liquids, and Everyday Objects* [2], centrados en la interacción con objetos cotidianos.

Para realizar identificación biométrica es necesario trabajar con frecuencias mayores a $5kHz$ como se observa en *A Multi-Frequency Current Source For Bioimpedance* [3], y se concluye que el cuerpo humano se comporta como un circuito RC a altas frecuencias en el apéndice A (*Brief introduction to bioimpedance*) del libro *Electrical impedance tomography: methods, history and applications* [4], lo que permite diferenciar entre distintos circuitos RC que dependen de la impedancia de cada cuerpo humano y su interacción con el entorno.

Capítulo 2

Transformada de Fourier

En el presente proyecto se desean obtener los valores de impedancia de un sistema a partir de la señal eléctrica que atraviesa ese sistema. Utilizaremos para ello una potente herramienta matemática utilizada comúnmente en tratamiento de señales, que nos ayudará a calcular las características de la señal que necesitemos. Esta herramienta es la Transformada de Fourier.

2.1. Definición

Las series de Fourier constituyen un tema clásico del análisis matemático, dada una función periódica se busca una serie trigonométrica 2.1.1 que la represente, esta es la serie de Fourier de la función, donde a_0, a_k y b_k son los coeficientes de Fourier que la representan.

$$\frac{a_0}{2} + \sum_{k=1}^{N} (a_k cos kx + b_k sin kx) \tag{2.1.1}$$

Las series de Fourier representan funciones periódicas definidas en un intervalo de la recta real, para representar funciones definidas en toda la recta y no periódicas se utiliza la transformada de Fourier. La transformada de Fourier es una operación matemática que permite transformar una función del dominio temporal al dominio de la frecuencia y viceversa, permite trabajar en frecuencia siendo este el objetivo de nuestro proyecto.

A continuación se indican las condiciones, la definición formal y algunas de las propiedades más importantes de esta transformación matemática:

- Condiciones para que exista la Transformada de Fourier:
  - $f(t)$ es absolutamente integrable

$$\int_{-\infty}^{\infty} |f(t)| dt < \infty (\text{Converge}) \tag{2.1.2}$$

- f(t) existe y es continua por intervalos [a,b] finito

$$\lim_{t \to t_0^{\pm}} f(t) \text{finito} \tag{2.1.3}$$

$$\lim_{t \to t_0^{\pm}} f(t) = \frac{f(t^+) + f(t^-)}{2} \tag{2.1.4}$$

- Definición:

  Para una función en el dominio del tiempo $f(t)$ real o compleja, con variable real t, se define la transformada de Fourier como:

$$\mathbb{F}[f(t)] = F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} \partial t \tag{2.1.5}$$

  Igualmente, tenemos la función inversa de Fourier:

$$\mathbb{F}^{-1}[F(\omega)] = f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{+j\omega t} \partial \omega \tag{2.1.6}$$

  De forma que se cumple

$$\mathbb{F}^{-1} \underbrace{[\mathbb{F}[f(t)]]}_{F(\omega)} = f(t) \tag{2.1.7}$$

$$\mathbb{F} \underbrace{[\mathbb{F}^{-1}[F(\omega)]]}_{f(t)} = F(\omega) \tag{2.1.8}$$

  Algunas de las propiedades de la Transformada de Fourier están definidas en la figura 2.1.

  Las demostraciones de las propiedades se pueden consultar en [5, 6, 7].

Linealidad	$\mathbb{F}\left[\alpha f(t) + \beta g(t)\right] = \alpha F(\omega) + \beta G(\omega)$
Dualidad	$\mathbb{F}[f(t)] = F(\omega) \rightarrow \mathbb{F}[F(t)] = 2\pi f(-\omega)$
Cambio de escala	$\mathbb{F}[f(at)] = \frac{1}{\lvert a\rvert} F\left(\frac{\omega}{a}\right)$
Transformada de la conjugada	$\mathbb{F}[f^*(t)] = F^*(-\omega)$
Translacion en el tiempo	$\mathbb{F}[f(t - t_0)] = e^{-j\omega t_0} F(\omega)$
Translacion en frecuencia	$\mathbb{F}[e^{+j\omega_0 t} f(t)] = F(\omega - \omega_0)$
Derivacion en el tiempo	$\mathbb{F}\left[\frac{\partial^n f(t)}{\partial t^n}\right] = (j\omega)^n F(\omega)$
Derivacion en la frecuencia	$\mathbb{F}\left[(-jt)^n f(t)\right] = \frac{\partial^n F(\omega)}{\partial \omega^n}$
Transformada de la integral	$\mathbb{F}\left[\int_{-\infty}^{t} f(\tau)\partial\tau\right] = \frac{F(\omega)}{j\omega} + \pi F(0)\delta(\omega)$
Transformada de la Convolucion	$\mathbb{F}\left[f(t) * g(t)\right] = \mathbb{F}\left[\int_{-\infty}^{\infty} f(\tau) g(t-\tau)\partial\tau\right] = F(\omega)G(\omega)$
Teorema de Parseval	$\int_{-\infty}^{\infty} \lvert f(t)\rvert^2 \partial t = \frac{1}{2\pi}\int_{-\infty}^{\infty} \lvert F(\omega)\rvert^2 \partial\omega$

Figura 2.1: Principales propiedades de la Transformada de Fourier.

2.2. Transformada Discreta de Fourier

La transformada de Fourier que se ha presentado en el apartado anterior está definida para sistemas continuos en el tiempo, pero en este proyecto se va a trabajar con señales muestreadas para poder procesarlas con un PC o un microprocesador Arduino, del que se habla en el capítulo 3, y mediante la herramienta Matlab, por esa razón no es posible aplicar la definición de Transformada de Fourier anterior.

Por tanto una vez definida la Transformada de Fourier, se presenta la transformada discreta de Fourier o DFT (del inglés, discrete Fourier transform) en la que las operaciones involucradas son sumas y multiplicaciones. Al igual que la transformada de Fourier, transforma una función matemática en otra, obteniendo una representación en el dominio de la frecuencia. Pero a diferencia de la Transformada de Fourier, la DFT requiere que la función de entrada sea una secuencia de datos discreta y de duración finita. Dichas secuencias se generan a partir del muestreo de una función continua, en nuestro caso particular esas señales son las tensiones muestreadas en los puertos de entrada del dispositivo Arduino y que se almacenan en secuencias de N datos [8, 5].

Por tanto la DFT transforma un conjunto de N muestras $x(nT)$ de la señal adquirida con el sistema de adquisición de datos (Arduino) en el mismo número de muestras discretas en frecuencia, $X(kF)$. La DFT es invertible y se habla de la IDFT (Inverse DFT). Las fórmulas que definen estas transformadas son [9]:

$$X(kF) = \sum_{n=0}^{N-1} x(nT) e^{\frac{-i2\pi}{N} nk} \quad \text{donde } F = \frac{1}{N \cdot T} \tag{2.2.1}$$

$$x(nT) = \frac{1}{N} \sum_{k=0}^{N-1} X(kF) e^{\frac{i2\pi nk}{N}} \tag{2.2.2}$$

En el Apéndice A se explica un método para llegar a este resultado [10].

Una descripción simple de estas ecuaciones es que los números complejos X_k representan la amplitud y fase de diferentes componentes sinusoidales de la señal de entrada X_n. La DFT calcula X_k a partir de X_n, mientras que la IDFT muestra cómo calcular X_n como la suma de componentes sinusoidales $\frac{1}{N} X_k e^{\frac{2\pi ikn}{N}}$ con una frecuencia de $\frac{k}{N}$ ciclos por muestra. Escribiendo las ecuaciones de este modo, estamos haciendo un uso extensivo de la fórmula de Euler para expresar sinusoides en términos de exponentes complejas:

$$e^{ix} = \cos(x) + i \sin(x). \tag{2.2.3}$$

Para realizar estos cálculos con un procesador, se requiere una gran cantidad de tiempo de cálculo, motivo por el cual se utliza la transformada rápida de Fourier o FFT (del inglés, Fast Fourier Transform).

2.3. Transformada Rápida de Fourier (FFT)

Se trata de un algoritmo que permite calcular la DFT y la IDFT. El algoritmo pone algunas limitaciones en la señal y en el espectro resultante.

Mientras la DFT requiere de N^2 multiplicaciones, si N es par se puede organizar el cálculo de manera que se realicen dos DFT y una combinación adecuada de ellas; esto reduce el número de multiplicaciones a aproximadamente la mitad, aplicando este proceso sucesivamente se reduce considerablemente el número de multiplicaciones [5].

La FFT se programa en un procesador que realiza todas las operaciones. Toma N muestras de la señal que se almacenan en memoria (ventana temporal), procesándose posteriormente para obtener el resultado deseado. El problema computacional para calcular la DFT consiste en calcular la secuencia $X(kF)$ a partir de la secuencia de datos $x(nT)$.

$$X(k) = \sum_{n=0}^{M-1} x(n) W_M^k \tag{2.3.1}$$

Donde $W_M^k = e^{\frac{i2\pi nk}{N}}$ para $k = 0, 1, ...M-1$.

Dados los datos $x(n), n = 1, ..., M$, se puede calcular directamente la secuencia $X(kF)$ a partir de la fórmula que define la DFT. Hay que calcular M elementos $X(kF)$, con M sumas complejas para cada elemento. La complejidad del algoritmo directo crece con M^2. Resulta un algoritmo ineficiente porque no explota las propiedades de simetría y periodicidad del factor de fase W_M^k :

$$W_M^{k+\frac{M}{2}} = -W_M^k$$
$$W_M^{k+M} = W_M^k$$

El algoritmo FFT consigue realizar el mismo cálculo pero su complejidad crece con M logM. En realidad no hay un único algoritmo FFT sino varios. Por concretar, vamos a comentar el primero que se conoció, denominado algoritmo de FFT de Cooley-Tukey (o algoritmo FFT-CT) [11, 12]. Este algoritmo exige que el número de muestras de la señal sea potencia de 2, lo que realmente no es una limitación fundamental para el sistema de adquisición de datos.

Un algoritmo FFT-CT se organiza conforme a la iteración de dos operaciones. La primera operación se llama algoritmo de diezmado en el tiempo y

consiste en descomponer recursivamente la secuencia $x(n)$ en subsecuencias y calcular la DFT de cada subsecuencia. La segunda operación consiste en combinar las DFTs calculadas hasta obtener la DFT final de la secuencia $x(n)$ original [9].

En el Apéndice B se indica la demostración de este método.

2.4. La Transformada Rápida de Fourier (FFT) y Matlab

En este apartado se hará un breve inciso sobre cómo trabaja la herramienta Matlab, *Programa de cálculo matemático de la compañía Mathworks* [13], a la que dedicaremos parte del capítulo 6, que se usa en este proyecto para el tratamiento de los datos y para obtener la FFT.

Matlab es el lenguaje de alto nivel y el entorno interactivo utilizado por millones de ingenieros y científicos en todo el mundo. Le permite explorar y visualizar ideas, así como colaborar interdisciplinarmente en procesamiento de señales e imagen, comunicaciones, sistemas de control y finanzas computacionales [13].

Se trata pues de una herramienta matemática que, entre otras cosas, permite implementar la FFT como una función que trate los datos muestreados y hacer un análisis frecuencial de la señal de entrada. Asimismo, Matlab incluye extensas librerías de funciones matemáticas y entre ellas incluye la función FFT en varias modalidades; se elegirá la que se adapte mejor a nuestro problema.

Capítulo 3

Plataforma Arduino

Se usará la plataforma Arduino como una tarjeta de adquisición de datos de bajo coste y dimensiones para tomar las muestras necesarias para el cálculo de la FFT comentada en el apartado 2.4.

3.1. ¿Qué es Arduino?

Arduino es una plataforma de electrónica abierta (open-source) para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores a través de los pines de salida. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectarlo a un ordenador (ser autónomos), si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software en ejecución (p.ej. Flash, Processing, MaxMSP, etc).

Figura 3.1: Placa Arduino UNO.

Las placas se pueden ensamblar a mano o encargarlas preensambladas y el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta [14, 15].

3.2. Hardware

La plataforma Arduino dispone de un buen número de placas diferentes [16]:

- Arduino Uno
- Arduino Leonardo
- Arduino Due
- Arduino Yún
- Arduino Micro
- Arduino Mega ADK
- Arduino Ethernet
- Arduino Mega 2560
- Arduino Robot
- Arduino Mini
- Arduino Nano
- LilyPad Arduino Simple y SimpleSnap
- LilyPad Arduino
- Arduino Pro

En nuestro caso vamos a centrarnos en la placa Arduino UNO que tenemos a nuestra disposición para realizar este proyecto.

La placa Arduino UNO está basado en el chip ATmega328, un chip microcontrolador creado por Atmel y pertenece a la serie megaAVR.

El Atmega328 AVR 8-bit es un circuito integrado de alto rendimiento que está basado un microcontrolador RISC, combinando 32 KB ISP flash, una memoria con la capacidad de lectura y escritura simultáneas, 1 KB de memoria EEPROM, 2 KB de SRAM, 23 lineas de E/S de propósito general, 32 registros de proceso general, tres temporizadores flexibles/contadores con modo de comparación, interrupciones internas y externas, programador de modo USART, una interfase serial orientada a byte de 2 cables, SPI puerto serial, 6-canales 10-bit Conversor A/D , "watchdog timer"programable con oscilador interno, y cinco modos de ahorro de energía seleccionables por software. El dispositivo opera entre 1.8 y 5.5 voltios [17].

La placa Arduino UNO tiene 14 pines entrada/salida digitales (de los cuales 6 pueden ser usados como salidas PWM (Pulse Width Modulation) y 6 entradas analógicas, un oscilador resonador cerámico, una conexión USB, conector de alimentación, una cabecera ICSP, y un botón de reset [18].

3.2.1. Alimentación

El Arduino UNO puede ser alimentado a través de la conexión USB o con un suministro de energía externo. La fuente de energía se selecciona automáticamente. La alimentación externa (no USB) puede venir o desde un adaptador AC-DC o desde una batería.

La placa puede operar con un suministro externo de 6 a 20 voltios. Si es suministrada con menos de 7 V, sin embargo, el pin de 5 V puede suministrar menos de cinco voltios y la placa podría ser inestable. Si usa más de 12 V, el regulador de tensión puede sobrecalentarse y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes [16, 18]:

- VIN. La entrada de tensión a la placa Arduino cuando está usando una fuente de alimentación externa (al contrario de los 5 voltios de la conexión USB u otra fuente de alimentación regulada). Se puede suministrar tensión a través de este pin, o, si suministra tensión a través del conector de alimentación, acceder a él a través de este pin.
- 5V. El suministro regulado de energía usado para alimentar al microcontrolador y otros componentes de la placa. Este puede venir o desde VIN a través de un regulador en la placa, o ser suministrado por USB u otro suministro regulado de 5 V.
- 3V3. Un suministro de 3.3 V generado por el chip FTDI de la placa. La corriente máxima es de 50 mA.
- GND. Pines de tierra.
- IOREF. Este pin provee la tensión de referencia con la que opera el microcontrolador.

3.2.2. Memoria

El ATmega168 tiene 16 KB de memoria Flash para almacenar código (de los cuales 2 KB se usa para el bootloader). Tiene 1 KB de SRAM y 512 bytes de EEPROM (que puede ser leída y escrita con la librería EEPROM [20]).

3.2.3. Entrada y Salida

Cada uno de los 14 pines digitales del Diecimila puede ser usado como entrada o salida, usando funciones *pinMode(), digitalWrite() y digitalRead()* [19].

Operan a 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia interna *pull-up* (desconectada por defecto) de 20-50 KOhms. Además, algunos pines tienen funciones especiales [16, 18]:

- **Serial: 0 (Rx) y 1 (Tx).** Usados para recibir (Rx) y transmitir (Tx) datos TTL en serie. Estos pines estan conectados a los pines correspondientes del chip FTDI USB-a-TTL Serie.

- **Interruptores externos: 2 y 3.** Estos pines pueden ser configurados para disparar un interruptor en un valor bajo, un margen creciente o decreciente, o un cambio de valor.

- **PWM: 3, 5, 6, 9, 10 y 11.** Proporcionan salida PWM de 8 bits con la función *analogWrite()*.

- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** Estos pines soportan comunicación SPI, la cual, aunque proporcionada por el hardware subyacente, no está actualmente incluida en el lenguaje Arduino.

- **LED: 13.** Hay un LED empotrado conectado al pin digital 13. Cuando el pin está a valor HIGH, el LED está encendido, cuando el pin está a LOW, está apagado.

  El ArduinoUNO tiene 6 entradas analógicas numeradas de A0 a A5, cada una de las cuales proporciona 10 bits de resolución (1024 valores diferentes). Por defecto miden 5 voltios desde tierra, aunque es posible cambiar el valor más alto de su rango usando el pin ARF y algún código de bajo nivel. Además, algunos pines tienen funcionalidad especializada:

- **I^2C: 4 (SDA) y 5 (SCL).** Soportan comunicación I^2C (TWI) usando la libreria *Wire*.

  Hay otro par de pines en la placa:

- **AREF.** Voltaje de referencia para las entradas analógicas. Usado con *analogReference()*.

- **Reset.** Pone esta línea a LOW para resetear el microcontrolador. Típicamente usada para añadir un botón de reset a dispositivos que bloquean a la placa principal.

3.2.4. Comunicación

La placa ArduinoUNO tiene un numero de infraestructuras para comunicarse con un ordenador, otro Arduino, u otros microcontroladores. El ATmega168 provee comunicación serie UART TTL (5 V), la cual está disponible en los pines digitales 0 (Rx) y 1 (Tx). Un ATmega16U2 en la placa canaliza esta comunicación serie al USB y proporciona un puerto de comunicación virtual al software del ordenador. El software Arduino incluye un monitor serie que permite a datos de texto simple ser enviados a y desde la placa Arduino.

Una libreria *SoftwareSerial* permite comunicación serie en cualquiera de los pines digitales del ArduinoUNO. El ATmega168 también soporta comunicación I^2C(TWI) y SPI. El software Arduino incluye una librería Wire para simplificar el uso del bus I^2C. Para usar la comunicación SPI, consultar el esquema del ATmega168 [21, 16, 18].

3.3. Software

El ArduinoUNO puede ser programado con el software Arduino, será el método empleado en este proyecto [22].

El ATmega168 del ArduinoUNO viene con un bootloader pregrabado para actualizar el código sin usar un programador hardware externo. Se comunica usando el protocolo original STK500. También se puede saltar el bootloader y programar el ATmega168 a través de la cabecera ICSP (In-Circuit Serial Programming).

Una vez instalado el software Arduino, y conectada la placa al PC por puerto USB, se crea el puerto serie virtual comentado anteriormente. Se selecciona ese puerto en el programa Arduino que se muestra en la Figura 3.2 y ya se puede cargar código creado o monitorizar los datos enviados por el puerto serie cuando la placa ejecuta el programa cargado.

3.4. Modulación por Ancho de Pulsos y programación en Arduino

Se va a utilizar Arduino como tarjeta de adquisición de datos (DAQ), para ello se crea un código cuya función es:

- 1º. Crear una señal PWM (Pulse Width Modulation) de frecuencia F(Hz) y Duty Cycle (DC) = 50 % (no confundir con corriente directa o continua DC).

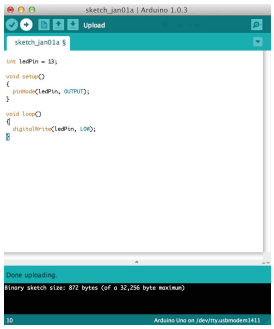

Figura 3.2: Entorno Arduino.

- 2º. Leer los pines de entrada y almacenar los valores de la señal de entrada, salida y el tiempo.
- 3º. Envía por puerto serie los datos.

El código se presenta comentado, aunque con nociones básicas de lenguaje C o de lenguaje Arduino, es fácil de comprender para el lector. Se va a explicar con más detalle el punto 1º (como se crea la señal de entrada) debido a que se modifican parámetros del registro del microprocesador para poder seleccionar el modo PWM adecuado y con las carácterísticas de DC y frecuencia deseadas.

- PWM

  Una señal PWM o modulada por ancho de pulsos, se construye modificando el ciclo de trabajo de una señal periódica. Requiere de un circuito generador formado por un comparador y un oscilador de tipo diente de sierra y se utiliza normalmente para activar una carga o para transmitir información, véase figura 3.3.

  El ciclo de trabajo (DC) de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$$DC = \frac{\tau}{T}$$

  donde T es el periodo de la señal y τ es el tiempo que la señal es positiva, es decir el ancho de pulso [31].

  Normalmente en Arduino se puede programar un PWM con dos funciones:

  - **función *analogWrite(DC)***: genera un PWM de frecuencia constante, aproximadamente $500Hz$ donde puede variarse el DC

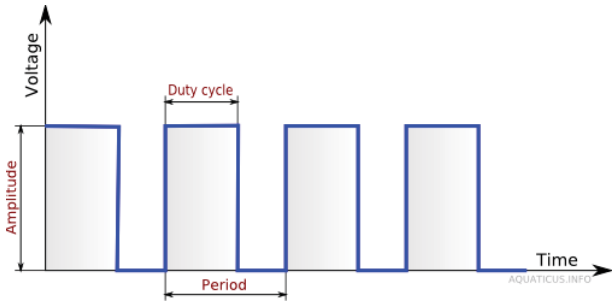

Figura 3.3: Ejemplo de señal PWM.

con valores entre 0-255 (de 0 %-100 % del DC). Esta función no es válida para nuestro proyecto porque se desea generar señales a distintas frecuencias para excitar las impedancias, como se expone en el capítulo 4.

- **Función *tone(f)***: genera una señal PWM de DC aproximadamente del 50 % y permite seleccionar la frecuencia deseada. Esta función está limitada por el tipo de PWM que genera, que es el tipo *normal*, existen varios tipos de PWM o formas de generarlo.

  Si se cambian los registros del microprocesador se puede acceder a otros tipos de PWM diferentes, con diferencias en *corrección de fase, corrección en frecuencia, modo*, que permiten obtener los distintos resultados en el proyecto.

El generador de PWM depende del reloj del sistema (system clock) que trabaja a una frecuencia de $16MHz$, el reloj del sistema está compuesto por varios relojes distintos, que sin entrar en detalle en ellos son:

- CPU clock
- I/O clock, que maneja los Timer/Counters(temporizadores y contadores) y los módulos de entrada/salida, que es lo que se persigue modificar en este apartado.
- Flash clock
- Asynchronous clock
- ADC clock

Para configurar el reloj se presenta el system clock prescaler, un parámetro configurable del reloj que permite dividir el system clock entre los distintos factores de la Tabla 3.4. Esto afecta a la frecuencia de la CPU, pero al usar este método se asegura que no aparezcan glitches en el reloj del sistema que afectan a las funciones temporales de Arduino

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock division factor
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	Reserved
1	0	1	0	Reserved
1	0	1	1	Reserved
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

Figura 3.4: Tabla de selección de clock prescaler [21].

(como *millis*() o *delay*()) y que aparecerían si se divide por cualquier número, el sistema está diseñado para usar los valores predeterminados.

El microchip además tiene 2 contadores de 8 bits (Counter0 y Counter2) y 1 de 16 bits (Counter1). Se usará el Counter2, que corresponde a los pines de salidas digitales 11 y 3. Se usará el 11 que afecta al canal A; véase figura3.5.

- 8-bit timer/Counter2 con PWM

  Una vez que se ha seleccionado el contador que se va a modificar, se buscan en el datasheet los registros que le afectan y se modifican:

  - Timer/Counter (TCNT2), 16 bits.
  - Output Compare Registers (OCR2A/B), 8 bits.
  - Input Capture Register (ICR2), 8 bits.
  - Timer/Counter Control Registers (TCCR2A/B), 8 bits.

  Para generar el PWM primero se configura el modo deseado. Se comentan en este documento dos de ellos, *Phase and frecuency* PWM *correct mode* (proporciona un PWM con corrección de fase y frecuencia de alta resolución), y *Fast* PWM *mode*, que genera PWM de alta frecuencia aproximadamente el doble que el modo anterior y nos permite aumentar el rango de nuestra medida.

timer	bits	channel	Arduino pin	Mega pin
timer0	8	A	6	13
timer0	8	B	5	4
timer1	16	A	9	11
timer1	16	B	10	12
timer2	8	A	11	10
timer2	8	B	3	9
timer3	16	A	-	5
timer3	16	B	-	2
timer3	16	C	-	3
timer4	16	A	-	6
timer4	16	B	-	7
timer4	16	C	-	8
timer5	16	A	-	44
timer5	16	B	-	45
timer5	16	C	-	46

Figura 3.5: Tabla con la relación pines y contadores [32].

En general su funcionamiento se basa en el contador TCNTx, que aumenta su valor en cada señal del reloj I/O y se compara con OCRxA o ICRx, que son los valores TOP o BOTTOM configurados.
Cuando se produce una coincidencia, se genera una interrupción y el generador de PWM genera una señal en el Output Compare Pin (OCxA/B) según el modo configurado en los bits Wave Generator Mode ($WGM13:0$) y Compare Output mode (COMxA1:0), que configura si el PWM es invertido o no invertido.

- **Fast PWM**
  En este modo el contador cuenta desde el valor BOTTOM hasta TOP y se reinicia desde BOTTOM de nuevo. Usa el método del lazo único.
  La resolución que se puede obtener puede ser de 8, 9 ó 10 bits según el modo configurado, o un valor según el valor de TOP:

$$R_{FPWM} = \frac{log(TOP+1)}{log(2)} \tag{3.4.1}$$

  Su frecuencia será:

$$f_{FPWM} = \frac{f_{clkI/O}}{ps \cdot (1+TOP)}$$

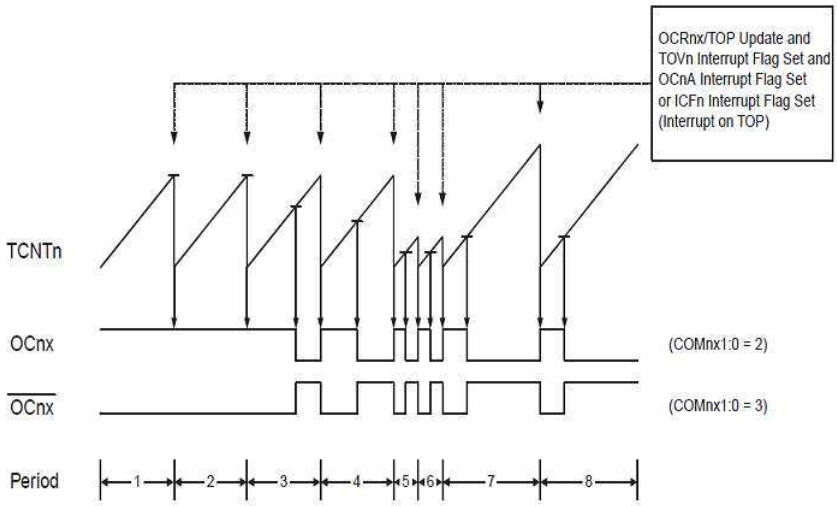

Figura 3.6: Diagrama Temporal del Fast PWM [21].

En la Figura 3.6 se encuentra el diagrama temporal del Fast PWM mode.

- **Phase and Frecuency Correct Mode**
  En este caso el contador cuenta desde BOTTOM hasta TOP y luego desde TOP hasta BOTTOM. Se basa en el método del doble lazo. La resolución depende del valor TOP que al igual que el Fast PWM se obtiene según la ecuación (ec.3.4.1). Su frecuencia será:

$$f_{PFCPWM} = \frac{f_{clkI/O}}{2 \cdot ps(TOP)}$$

  y su diagrama temporal puede verse en la Figura 3.7.
  En nuestro código para calcular la frecuencia deseada se despeja el valor TOP y se va modificando para cada barrido de frecuencia.

A continuación se incluyen las tablas 3.8, 3.9, 3.10, de configuración de los modos PWM comentados. Toda la información de este apartado se ha obtenido del datasheet del chip atmega328 [21], donde se puede encontrar más información sobre la configuración y funcionamiento de los registros y de otros modos PWM como el modo normal o el modo de corrección de fase.

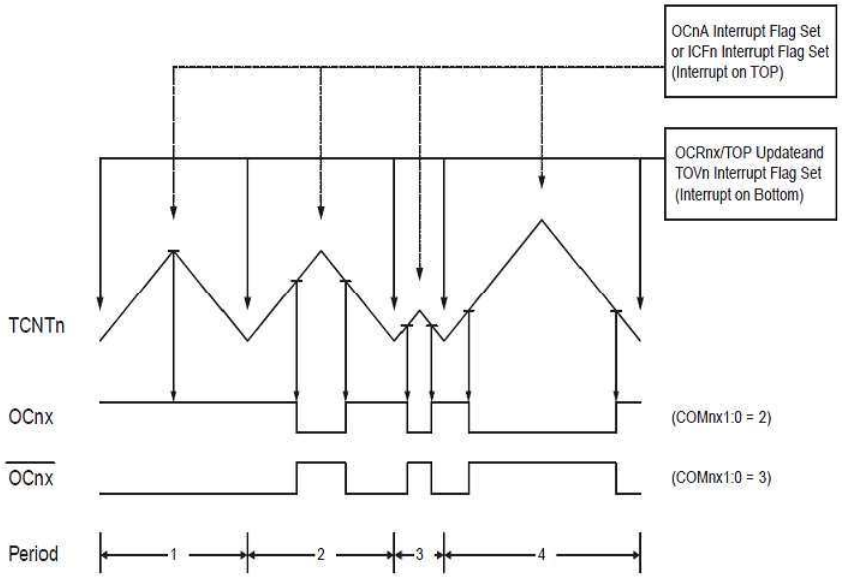

Figura 3.7: Diagrama Temporal del PFC PWM [21].

Como el registro OCRxA/B es de 8 bits cuando busca alcanzar frecuencias pequeñas el registro se desborda hasta alcanzar el valor deseado. Esto ocasiona el problema de que el PWM generado nunca corresponde con la frecuencia deseada. Para corregir este problema, hay que modificar el Prescaler (aumentando o disminuyendo su valor) de manera que el OCRx nunca rebase el valor de 255. Notese que cuando se requiere una frecuencia mayor de la que se puede alcanzar con el Prescaler actual hay que disminuirlo.
De igual modo, se debe ir ajustando el Prescaler del Convertidor Analógico Digital (ADC) de 10 bits, que se encarga de leer el pin analógico de entrada, para que pueda leer a mayor o menor frecuencia, buscando la máxima resolución posible. El datasheet indica que no hay pérdida de resolución considerable por debajo de $1MHz$ de frecuencia de muestreo; esto corresponde a configurar el ADC Prescaler en 16. Experimentalmente se ha comprobado que es cierto; si sigue aumentando la frecuencia de muestreo y la resolución es menor a 8 bits debido a este aumento de la frecuencia se obtienen valores erróneos en la lectura.

En el Apéndice C se muestra el código empleado comentado.

Table 15-4. Waveform Generation Mode Bit Description[1]

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Figura 3.8: Tabla de bits para configurar los PWM [21].

Table 15-2. Compare Output Mode, Fast PWM[1]

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at BOTTOM (inverting mode)

Figura 3.9: Tabla de bits para configurar COM en Fast PWM [21].

Table 15-3. Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM[1]

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 11: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match when up-counting. Set OC1A/OC1B on Compare Match when downcounting.
1	1	Set OC1A/OC1B on Compare Match when up-counting. Clear OC1A/OC1B on Compare Match when downcounting.

Figura 3.10: Tabla de bits para configurar COM en PFC PWM [21].

Capítulo 4

Medida de impedancia eléctrica

La impedancia eléctrica es la oposición que experimenta un dispositivo o circuito eléctrico o electrónico a la circulación de corriente eléctrica. Es una propiedad característica de los medios materiales en general y de los elementos de circuito en particular, que describe su comportamiento con respecto a la circulación de una corriente eléctrica a través de ellos cuando se aplica una determinada diferencia de potencial eléctrico entre sus extremos [23].

La amplitud y frecuencia del potencial eléctrico aplicado tienen influencia sobre la respuesta eléctrica del elemento considerado y así aparece el concepto de espectrometría de impedancia eléctrica, técnica que busca determinar el valor de la impedancia eléctrica y su variación en función de la frecuencia y amplitud de la señal aplicada.

Cuando una señal se transmite a través de un medio material, se pueden producir alteraciones tanto en la amplitud como en la fase. Por ello, una función que relaciona las señales incidente y transmitida, como es la impedancia, tiene dos componentes que dan cuenta de estos parámetros. Tiene por tanto carácter vectorial y se podrá expresar como un número complejo [25].

Se puede expresar una impedancia como $Z = R + jX$ donde R representa la parte real y jX la parte imaginaria, siendo R (resistencia o parte resistiva) y X (reactancia o parte reactiva) números reales, o como un fasor $Z = |Z|\angle\phi$, según el valor del argumento(ϕ) la impedancia será :

- $\phi = 0$ impedancia resistiva
- $\phi < 0$ impedancia capacitiva
- $\phi > 0$ impedancia inductiva

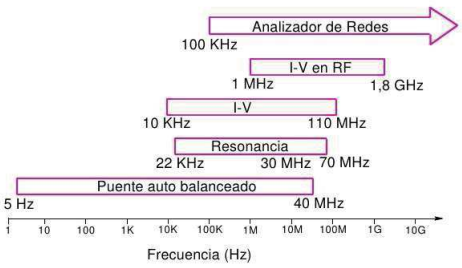

Figura 4.1: Técnicas de medida en función de la frecuencia.

4.1. Medida de la impedancia

La impedancia, como se ha visto en el apartado anterior, tiene carácter vectorial y se expresa como un número complejo. Hay distintos métodos para medir la impedancia de un elemento dependiendo de la frecuencia, tal y como se muestra en la Figura 4.1 [23]:

- **Puente de impedancias**
- **Divisor de impedancia**
- **Otros:**resonancia,I-V...

4.1.1. Puente de impedancias

Los puentes utilizan el método de medida de comparación por balanceo y diferencial. Para este caso se hace que un indicador (un galvanómetro o un amperímetro) proporcione una condición de deflexión nula o conocida cuando se compara la cantidad medida con la cantidad patrón. Los puentes se emplean para medida de resistencias, capacidades, inductancias, factor de calidad, ángulo de pérdidas, etc [26].

Un puente de corriente alterna se puede representar por dos ramas como se muestra en la Figura 4.2.

Mediante una impedancia variable, se equilibran las dos ramas del puente haciendo que la intensidad que atraviesa el galvanómetro sea 0, de manera que una vez que $I_g = 0$ se dice que el puente está equilibrado y se cumplen las siguientes relaciones entre los módulos y argumentos de las impedancias:

$$|Z_1| \cdot |Z_3| = |Z_2| \cdot |Z_4| \tag{4.1.1}$$

$$\phi_1 + \phi_3 = \phi_2 + \phi_4 \tag{4.1.2}$$

Estas ecuaciones permiten el cálculo de cualquier impedancia desconocida facilmente conociendo las otras 3 impedancias patrón y ajustando el

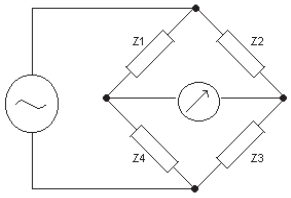

Figura 4.2: Puente de corriente alterna.

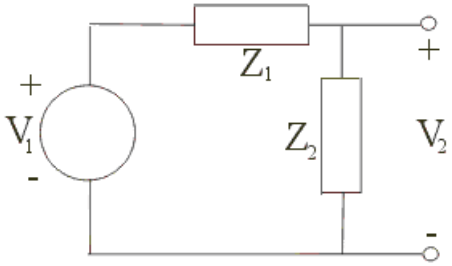

Figura 4.3: Montaje divisor de impedancia.

galvanómetro o amperímetro de forma que la intensidad sea nula a través de él, aunque normalmente se usa en sensores para medir la variación de una magnitud (temperatura, presión, etc.).

4.1.2. Divisor de impedancia

Los divisores de impedancia tienen como misión establecer una comparación entre la señal de referencia y la señal a medir. Para resolver el proceso de medida puede emplearse un montaje sencillo mediante divisor de impedancias como el que se muestra en la Figura 4.3 [26].

Si se supone que Z_1 y la señal de entrada V_1 son conocidas, se puede medir la señal de salida V_2 y calcular el valor de Z_2 fácilmente despejando de 4.1.3:

$$\frac{V_2}{V_1} = \frac{Z_2}{Z_1 + Z_2} \tag{4.1.3}$$

$$Z_2 = \frac{Z_1 V_2}{V_1 - V_2} \tag{4.1.4}$$

donde V_1, V_2, Z_1 y Z_2 son números complejos.

A partir de los resultados anteriores se concluye que para medir la impedancia Z_2 debemos excitar el circuito con corriente alterna (CA, AC en inglés) y medir la señal de salida.

4.1.3. Otros: resonancia,I-V.

- **Resonancia:** utilizando las propiedades de resonancia de las ondas electromagnéticas permite realizar barridos en frecuencia y obtener el factor de calidad (Q) del circuito, generalmente condensadores y bobinas. El medidor de Q sintoniza un condensador hasta que queda en resonancia con el circuito y eso permite mediante ciertas relaciones conocer la impedancia y la Q del circuito medido, más información en *Medidores de Impedancia y parámetros de componentes pasivos* [23] y en *Resonancia de ondas electromagnéticas. Medida de la permitividad dieléctrica de líquidos polares* [24].

- **I-V:** utiliza una sonda I-V para medir la caída de tensión y la fuerza electromagnética inducida por la corriente y con esa información calcula la impedancia [23].

4.2. Análisis de circuitos de corriente alterna y cálculo de la impedancia

Un circuito lineal de CA es un sistema eléctrico compuesto de componentes lineales, como pueden ser resistencias, inductancias y capacitancias, además de fuentes sinusoidales.

Estos sistemas trabajan en régimen sinusoidal. Esto quiere decir que en la ecuación diferencial utilizada para calcular la corriente en función del tiempo la solución homogénea se extingue rápidamente (transitorio), por lo que solo interesa la solución particular. Esta última será sinusoidal si la excitación lo es, puesto que el circuito es lineal. Por tanto, sabemos que en un circuito de CA todas las corrientes y voltajes van a ser sinusoides, con la misma frecuencia w (siendo w la frecuencia de la excitación), aunque distinta amplitud y desfase [27].

Suponemos que la señal de entrada a nuestro circuito es una señal consenoidal $v_1(t) = V_{p1}cos(wt+\theta)$, que puede escribirse como el fasor $V_1 = V_{p1}e^{j\theta}$. Como es lineal, a la salida del divisor de tensión de la figura 4.3 mediremos la señal $v_2(t) = V_{p2}cos(wt + \theta + \phi) = V_2 = V_{p2}e^{j\theta+\phi}$, donde ϕ es el desfase introducido en la señal debido a la impedancia del circuito.

Conocido el valor de $Z_1 = R_1 + jX_1$, (para simplificar los cálculos se tomará $X_1 = 0$ de forma que Z_1 es una resistencia pura) se puede calcular fácilmente el valor de la impedancia Z_2 que queremos hallar en función de la frecuencia de la señal de entrada, como ya se comentó en la sección 4.

$$Z_2 = \frac{R_1 \cdot V_2}{V_1 - V_2} = \frac{R_1 \cdot V_{p2}e^{j\theta+\phi}}{V_{p1}e^{j\theta} \cdot V_{p2}e^{j\theta+\phi}} \tag{4.2.1}$$

La operación no es un cálculo trivial de realizar a mano, pero utilizando las propiedades de los números complejos, la forma fasorial para multiplicaciones y divisiones, y la forma binómica para sumas y restas se puede hallar el resultado fácilmente pasando de una a otra, aunque en nuestro caso usaremos la herramienta matemática Matlab para ayudarnos en las operaciones.

El resultado será un numero complejo del tipo $Z_2 = R_2 + jX_2$ o $Z_2 = |Z_2|\angle\phi_2$.

Capítulo 5

Clasificación de impedancias

Como vimos en el apartado anterior 4.2, se puede calcular el valor de una impedancia Z para el valor de frecuencia de la señal de entrada. No obstante la impedancia tiene una notable dependencia con esa frecuencia. Por tanto es posible clasificar las distintas impedancias calculando su curva de variación a distintas frecuencias mediante un análisis frecuencial y el diagrama de Bode. Con este método se puede clasificar con mayor exactitud las impedancias medidas. Primero estudiaremos como afecta la frecuencia a la impedancia.

5.1. Variación de la impedancia con la frecuencia

Las impedancias de condensadores y bobinas varían con la frecuencia en los circuitos de corriente alterna, Comenzaremos con un estudio de circuitos simples RC y RL y estudiaremos su dependencia con la frecuencia.

5.1.1. Circuito RC y RL

- **Circuito RC**

  El condensador es un elemento que al someterse a una diferencia de potencial, genera una intensidad proporcional a la variación de la misma. La ecuación del modelo de este elemento es:

  $$\frac{dV(t)}{dt} = \frac{i(t)}{C} \tag{5.1.1}$$

  Se denomica C a la capacidad del condensador [30].

  La impedancia de un condensador en función de la frecuencia f (Hz) es $Z_C = \frac{-j}{Cw}$ donde $w = 2\pi f$ es la frecuencia angular y j representa la unidad imaginaria. En corriente alterna también es útil el empleo del concepto de reactancia. La reactancia capacitiva se define como $X_C = \frac{1}{Cw}$ y la impedancia puede expresarse $Z_C = -jX_C$.

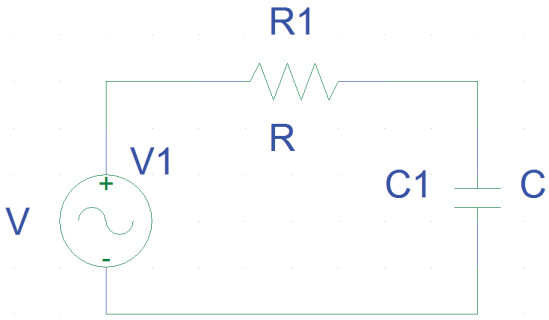

Figura 5.1: Circuito RC serie.

Sea un circuito RC serie como el que se muestra en la Figura 5.1[1], la impedancia del circuito será:

$$Z = R + Z_C = |Z|e^{-j\phi} \tag{5.1.2}$$

donde $|Z| = \sqrt{R^2 + \frac{1}{Cw}^2}$ y $tg(\phi) = \frac{1}{CwR}$.

- **Circuito RL**

  Una bobina es un elemento que al ser sometido a una diferencia de potencial genera una variación en la intensidad proporcional a la misma. Las bobinas almacenan energía eléctrica en forma de energía magnética. La ecuación del modelo es la siguiente:

$$V(t) = L\frac{di(t)}{dt} \tag{5.1.3}$$

  Siendo L la constante de autoinducción de la bobina.

  A continuación se procede a estudiar la impedancia de un circuito RL y cómo afecta una bobina a la impedancia del circuito, ya que su impedancia también varía en función de la frecuencia. La impedancia de una bobina de inductancia L en función de la frecuencia f (Hz) es $Z_L = jLw$, donde $w = 2\pi f$ es la frecuencia angular y j representa la unidad imaginaria. La reactancia inductiva se define como $X_L = Lw$ y la impedancia de la bobina será $Z_L = jX_L$.

  Asimismo, sea un circuito RL serie como el que se muestra en la Figura 5.2, la impedancia del circuito será:

$$Z = R + Z_L = |Z|e^{j\phi} \tag{5.1.4}$$

[1]Imágenes generadas con el programa PSpice versión de estudiante.

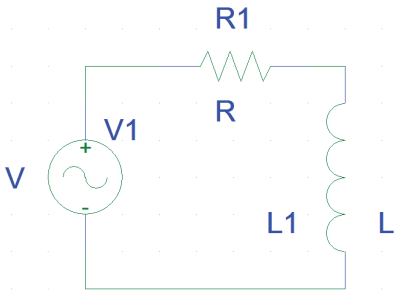

Figura 5.2: Circuito RL serie.

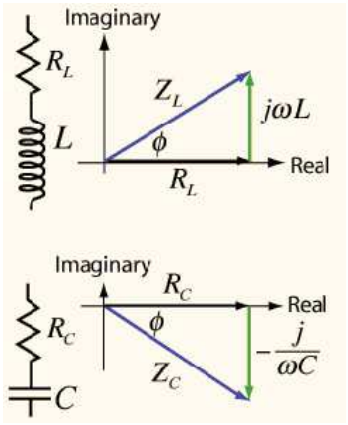

Figura 5.3: Representación gráfica de las impedancias en circuitos RL y RC.

donde $|Z| = \sqrt{R^2 + Lw^2}$ y $tg(\phi) = \frac{Lw}{R}$.

La impedancia del circuito que se mida en bornas del Arduino, se puede representar como suma de fasores, o se puede hacer la suma vectorial en el plano imaginario como el que se representa en la Figura 5.3. Se puede observar cómo afecta a la impedancia total del circuito el efecto de un condensador o de una bobina, y a partir de estos resultados se puede calcular cualquier impedancia como suma de impedancias en serie y paralelo de resistencias, condensadores y bobinas, y representar su impedancia en el plano imaginario, de manera que cada circuito poseerá una impedancia particular en función de la frecuencia de la corriente que lo alimente [29].

5.2. Diagrama de Bode

El diagrama de Bode (véase la Figura 5.4) es una representación gráfica que sirve para caracterizar la respuesta en frecuencia de un sistema. Es una herramienta que nos permite representar la variación de una magnitud (la impedancia en este caso) respecto a la frecuencia. En teoría de sistemas es el más utilizado para representar gráficamente la función de transferencia de un sistema y es muy utilizado también en análisis de circuitos.

Se representa mediante el conjunto de dos curvas:

- **Curva de Amplitud:** traza la amplitud de nuestro sistema, expresado en decibelios (dB), en función de w (en escala logarítmica), en nuestro caso particular $20 * log_{10}|Z(w)|$.
- **Curva de fase:** traza la fase de nuestro sistema en función de w (en escala logarítmica), en nuestro caso $arg(Z(w))$.

Obsérvese que tanto la amplitud como la fase se representan en función de la frecuencia expresada en escala logarítmica. Esto es debido a que, si los sistemas trabajan en un rango amplio de frecuencias, las escalas logarítmicas permiten una mejor representación. En el caso de escalas lineales, si se desea estudiar con detalle las frecuencias bajas, entonces las altas se salen del diagrama, y si se desea incluir las altas, las bajas salen muy comprimidas. Asimismo el logaritmo convierte productos en sumas. La escala logarítmica comprime más cuanto mayor es la frecuencia [30].

En este proyecto se hace uso del diagrama de Bode para obtener una relación de la impedancia de nuestro sistema (aquello que se coloque en las bornas del Arduino) con la frecuencia, y con esa información clasificar ese sistema.

5.2.1. Diagrama de Bode para circuitos RC y RL

Tal y como se comentó en el apartado 5.1.1, la impedancia de condensadores y bobinas varía con w. Se puede usar el programa PSpice [2] para simular la respuesta de estos circuitos y obtener el diagrama de Bode de cada uno de ellos o Matlab y la función *bode*.

- **Bode de circuito** RC

  Se conoce la impedancia del condensador $Z_C = \frac{1}{jwC}$. Por tanto la caída de tensión en bornas del condensador (véase Figura 5.1) es:

$$V_C = \frac{\frac{1}{jwC}}{R + \frac{1}{jwC}} V_1 = \frac{1}{jwCR + 1} V_1 \tag{5.2.1}$$

[2]SPICE Es un estándar internacional cuyo objetivo es simular circuitos electrónicos, PSpice pertenece a la empresa OrCAD

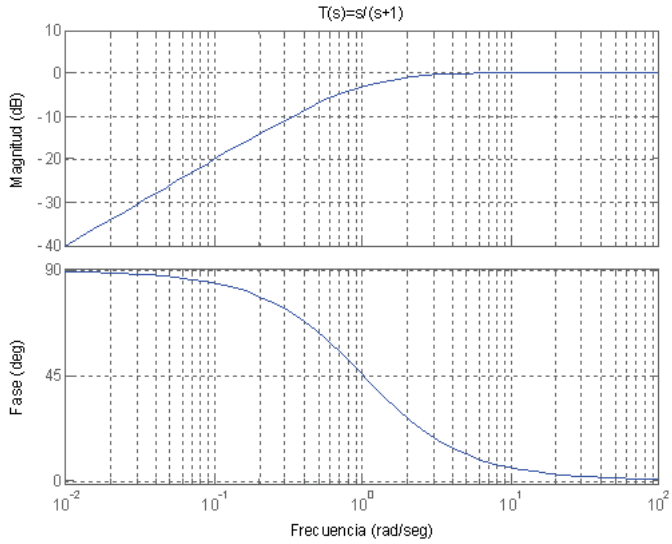

Figura 5.4: Ejemplo de un Diagrama de Bode.

Por teoría de sistemas [30] sabemos que la frecuencia compleja s es un número complejo que se puede sustituir $s = jw$ [3], de manera que el circuito tiene un polo estable, p, en $s = \frac{-1}{RC}$. En el diagrama de Bode esto se ve reflejado como una asíntota en 0 dB para frecuencias menores de p. Para frecuencias mayores de p el módulo tiende una recta de -20 $dB/década$ de pendiente.

La fase parte de 0° y decrece monotonamente hasta alcanzar $-90°$ como se aprecia en la Figura 5.5.

- **Bode de circuito** RL

  En el caso del circuito RL la impedancia $Z_L = Ljw$ y la caída de tensión es:

$$V_L = \frac{sL}{R + sL} V_1 = \frac{L}{R} \frac{s}{1 + \frac{sL}{R}} V_1 \tag{5.2.2}$$

  En este caso se obtiene un diagrama de Bode con una ganancia constante de $\frac{R}{L}$, un polo estable en $s = \frac{-R}{L}$ y un derivador que añade un módulo de 20 $dB/década$ tal que para $w = 0$ rad/s su valor sea de 0 dB y una fase constante de 90°.

Se puede concluir de la observación y las propiedades de los diagramas de Bode que cuando se intente clasificar una impedancia con parte capacitiva y

[3] al hacer esto se trabaja con la transformada de Fourier.

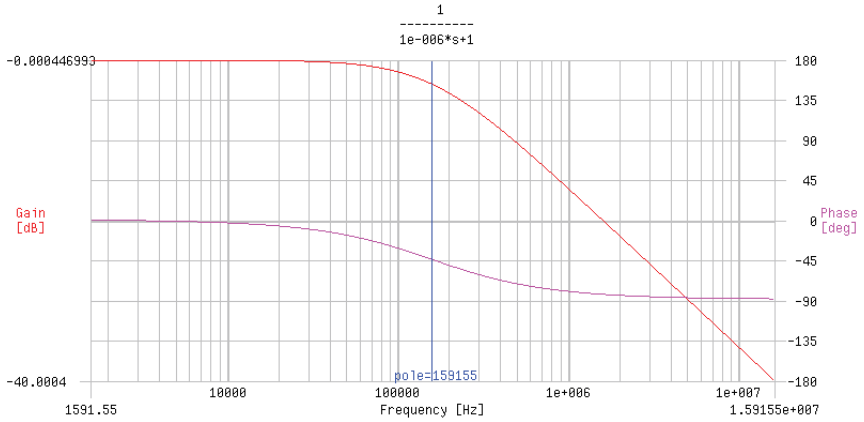

Figura 5.5: Diagrama de Bode del circuito RC.

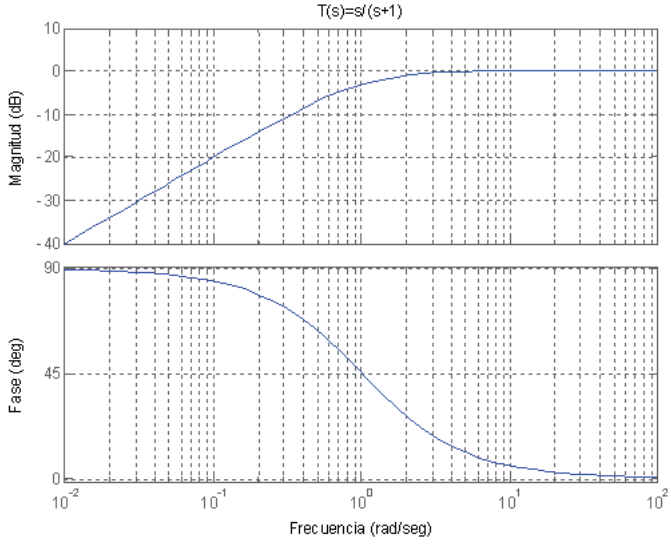

Figura 5.6: Diagrama de Bode del circuito RL.

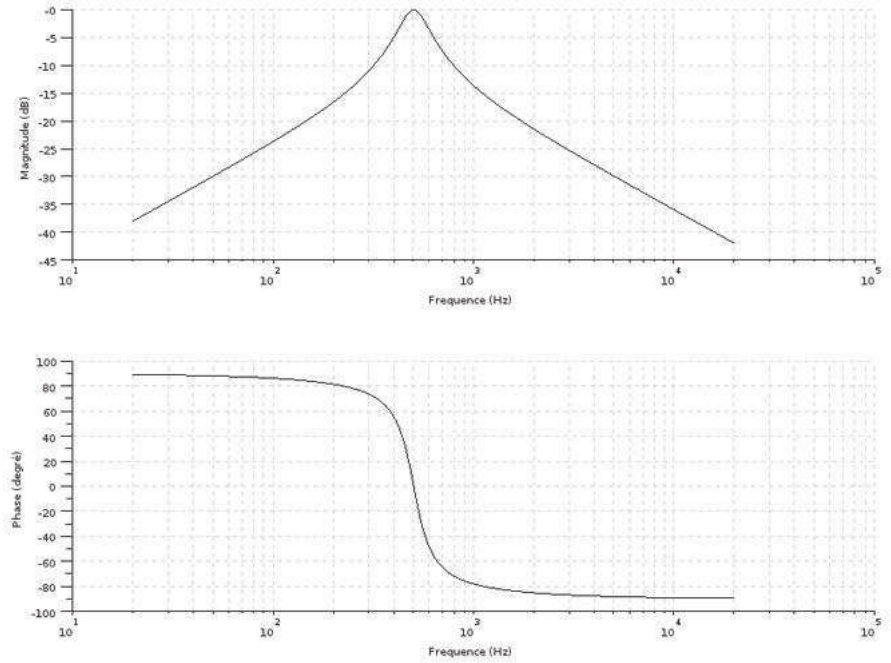

Figura 5.7: Diagrama de Bode del circuito RLC.

parte inductiva se obtendrá un diagrama suma de los anteriores (ponderado según el peso de la parte inductiva y capacitiva de la impedancia) como se observa en la Figura 5.7.

5.3. Método de clasificación

Una vez que ya se obtienen los diagramas de Bode correspondientes a los valores de impedancia, nuestro objetivo es clasificarla en torno a unos patrones previamente definidos.

Para realizar esta clasificación se va a utilizar el método de lógica borrosa o Fuzzy Logic, que nos indica que grado de pertenencia tiene la impedancia medida para cada patrón, y se asigna a aquel patrón con mayor grado de pertenencia.

5.3.1. Fuzzy Logic

Se trata de un concepto introducido por Lofti Zadeh. Combina los conceptos de la lógica y de los conjuntos de Lukasiewicz mediante la definición de grado de pertenencia.Es una lógica que permite valores imprecisos

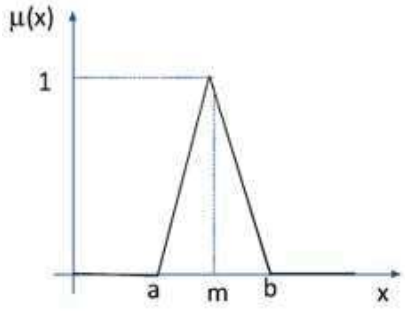

Figura 5.8: Función de pertenencia triangular

o intermedios para poder definir evaluaciones convencionales entre si/no o verdadero/falso.

Para entender este concepto correctamente se define el concepto de conjunto difuso.

- **Conjunto borroso**

  Un conjunto es una colección de elementos bien especificados con una propiedad común, los conjuntos se pueden definir entre otras formas por su función característica o función de pertenencia, si la función de pertenencia para un valor dado toma el valor 1, pertenece al conjunto y si toma el valor cero, no pertenece al conjunto.

  Para un conjunto borroso son posibles distintos grados de pertenencia, la función puede tomar cualquier valor en el intervalo [0,1], donde de nuevo 1 equivale a pertenencia perfecta y el valor cero indica que no pertenece al conjunto [33, 34].

En nuestro caso, si cada patrón de impedancia se considera un conjunto difuso, la impedancia que queremos clasificar tendrá un grado de pertenencia en el intervalo [0,1] para cada patrón.

Para clasificar las impedancias se utiliza un clasificador basado en la función borrosa de pertenencia triangular utilizada en *Fuzzy Matching Engine for Non-textual Authentication: a case study* [35].

- **Clasificador triangular borroso**

  Para calcular el grado de pertenencia (μ) de un elemento a cada patrón se requiere de una función de pertenencia que defina correctamente el grado de pertenencia en cada ocasión. En este proyecto se usará la función triangular de la Figura 5.8, definida por

$$\mu(x) = \begin{cases} 0 \text{ para } x \leq a \\ \frac{(x-a)}{(m-a)} \text{ para } a < x \leq m \\ \frac{(b-x)}{(b-m)} \text{ para } m < x \leq b \\ 0 \text{ para } x \geq b \end{cases} \tag{5.3.1}$$

donde a, b y m son parámetros.

Nuestro clasificador, una vez obtenido el grado de pertenencia de la impedancia medida, asignará esta a un patrón u otro dependiendo de a cual de los patrones corresponde mayor μ, tal y como se indica en el capítulo 7.

Capítulo 6

La herramienta matemática Matlab

Matlab es el nombre abreviado de MATrix LABoratory. Se trata de un programa para realizar cálculos numéricos con vectores y matrices. Puede también trabajar con números escalares, tanto reales como complejos, con cadenas de caracteres y con otras estructuras de información más complejas. Una de las capacidades más atractivas es la de realizar una amplia variedad de gráficos en dos y tres dimensiones. Matlab tiene también un lenguaje de programación propio.

Matlab es un gran programa de cálculo técnico y científico su lenguaje de programación siempre es una magnífica herramienta de alto nivel para desarrollar aplicaciones técnicas. Fácil de utilizar, dispone de un código básico y de varias librerías especializadas (toolboxes). Para más información consultar la página web [13] o alguno de los múltiples manuales como [28].

El programa de Matlab que se ha creado para el tratamiento de los datos, lee los datos adquiridos a través de Arduino y almacenados en un archivo de texto, calcula su diagrama de bode mediante el script explicado en el Apéndice D y clasifica ese diagrama, usando Fuzzy Logic, entre los distintos patrones que se encuentran almacenados en un archivo de extensión MAT, mediante el script del clasificador y finalmente los representa permitiendo visualizar que la clasificación es correcta. En la Figura 6.1 se muestra la interfaz de usuario del programa y el parte del script utilizado.

También se ha creado un script que genera el archivo MAT con los patrones, permite facilmente crear la base de datos de los patrones para despues usarlos en el clasificador, solo es necesario seleccionar el archivo de texto con los datos almacenados. Además este script genera gráficas con los patrones introducidos para verificar que es correcto el resultado almacenado en el archivo MAT que utilizará el clasificador, y además incluye un editor para las

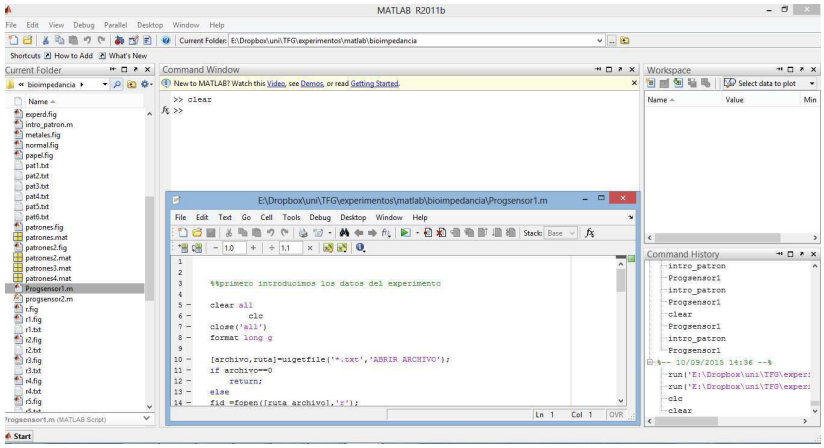

Figura 6.1: Interfaz de Matlab

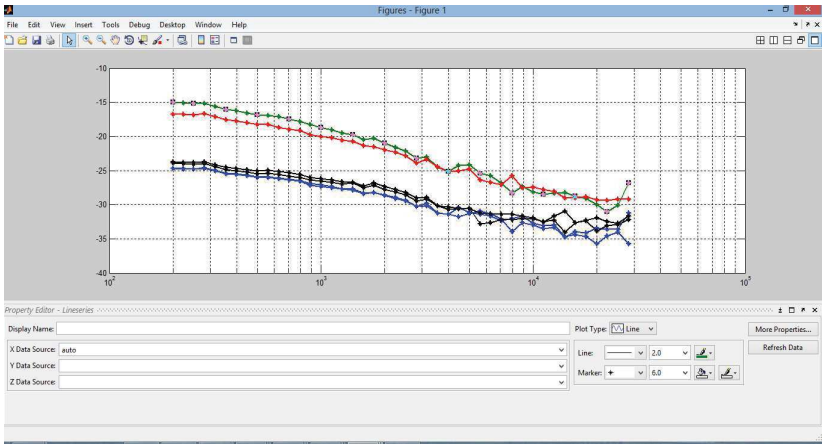

Figura 6.2: Editor gráfico de gráficos

gráficas que permite cambiar las propiedades de cada curva o añadir información como la leyenda o etiquetas, todo esto se puede además modificar desde la ventana de comandos o incluyendo el código en el script, véase la Figura 6.2.

En el Apéndice D se muestra el código empleado y comentado.

Capítulo 7

Experimentos, resultados y conclusiones

Este capítulo muestra la trayectoria seguida durante todo el proceso de investigación, los resultados y las conclusiones que se han obtenido a lo largo del proyecto.

El objetivo es aplicar las herramientas y conocimientos teóricos expuestos en los capítulos anteriores a experimentos con impedancias reales, demostrar si es posible conseguir el resultado esperado y las dificultades encontradas.

Los experimentos que se van a realizar son medidas de distintas impedancias conocidas, para su posterior clasificación con el clasificador borroso diseñado. En el capítulo 8 en lugar de medir impedancias conocidas se mide la bioimpedancia al interactuar con distintos objetos, para su posterior clasificación.

7.1. Experimentos

Durante el proceso de montaje y pruebas con la plataforma Arduino se fueron tomando pequeñas decisiones tales como utilizar un montaje en modo divisor de impedancia y leer tanto la señal de entrada como la de salida. Esto se observa en la Figura 7.1. Aunque se puede medir cualquier impedancia en las bornas de los pines de entrada, para obtener mejores resultados y simplificar los experimentos se usan impedancias conectadas a la protoboard, tal y como puede verse en las figuras 7.2 y 7.3.

Una vez hecho el montaje y cargado el programa en Arduino, comienza el experimento. En primer lugar, se lee en los pines analógicos $A4$ y $A5$ los valores de tensión, que se envían por puerto serie al PC donde se almacenan en un fichero de texto.

El experimento consiste en un barrido de frecuencias. Se ha configurado el código para que incremente la frecuencia en forma logarítmica y para que

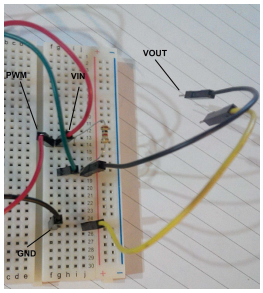

Figura 7.1: Montaje en modo divisor de tensión.

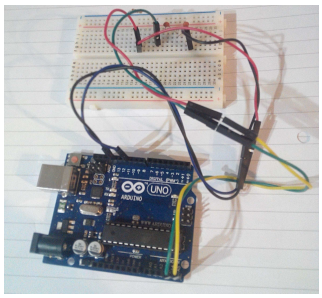

Figura 7.2: Conexión de impedancia con Arduino.

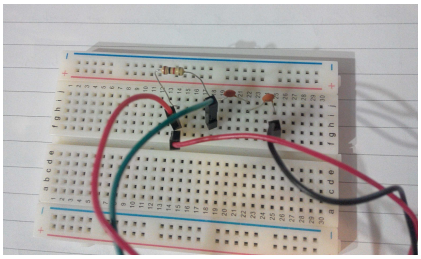

Figura 7.3: Ejemplo de medir una impedancia en modo divisor de tensión.

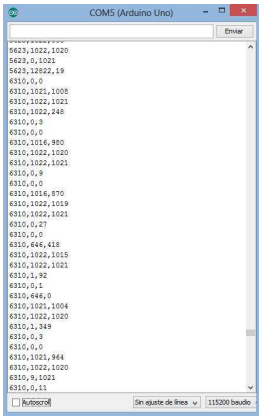

Figura 7.4: Transmisión de datos mostrados por la aplicación de Arduino.

se tomen 44 puntos o 44 frecuencias distintas. No obstante, este parámetro es configurable, podiendo tomar hasta 2000 puntos. Un valor alto de muestras requerirá mucho tiempo para reconstruir la señal medida y obtener la transformada de Fourier, por lo que se ha decidido leer el máximo número de muestras que permite Arduino en memoria, antes de volcar los datos al PC. En total para cada valor de frecuencia se leen 320 muestras para cada una de las 44 frecuencias en cada pin de entrada ($A4$ y $A5$), véase figura 7.4.

Una vez leídos todos los datos, el script de Matlab lee el archivo de texto y transforma los valores de entrada leídos de 0-1024 en nuestro voltaje (0-5V para el Arduino UNO). En la figura 7.5 se ve la reconstrucción de las dos señales; V_{in} corresponde al PWM de entrada y V_{out} es la caída de tensión en nuestra impedancia (capacitiva en este caso). El tiempo no es necesario debido a que la técnica de la FFT no precisa de información sobre el mismo; sólo necesita los valores de las muestras. De hecho Arduino solo tiene un convertidor analógico-digital (CAD). Por lo que no es posible tomar dos muestras en el mismo tiempo de muestro. Por eso se tomó la decisión de medir ambas señales por separado y luego obtener su valor de amplitud, que es el valor necesario para crear el diagrama de Bode utilizando la transformada de Fourier.

A continuación calculamos la FFT de ambas señales, y se obtiene la gráfica de la Figura 7.6, donde aparecen las distintas amplitudes de la señal. Nótase los armónicos de la onda cuadrada. En nuestro caso se selecciona el armónico de mayor amplitud y se desechan los demás. Con esa información ya es posible obtener los valores de amplitud del diagrama de bode para cada frecuencia.

El siguiente paso es construir el diagrama de Bode, usando la fórmula

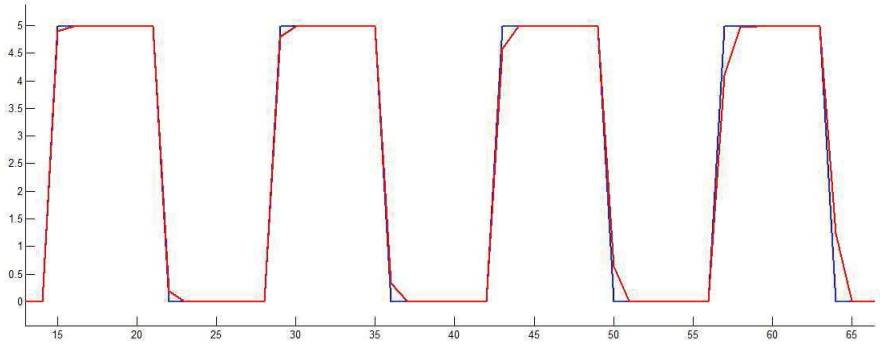

Figura 7.5: Señales leídas (V_{in}(azul) y V_{out}(rojo)).

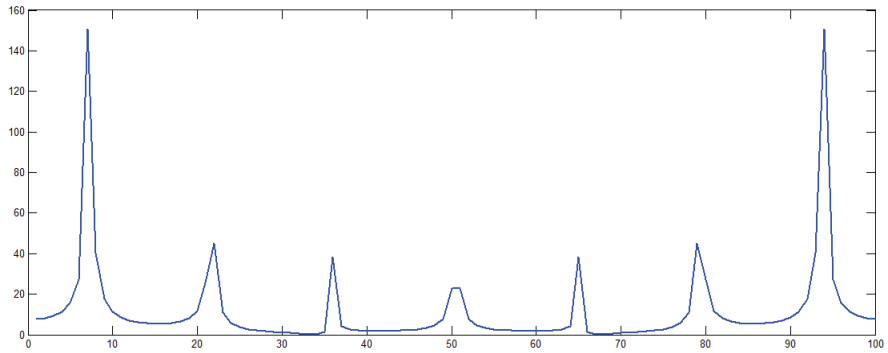

Figura 7.6: Espectro de frecuencia de V_{in}

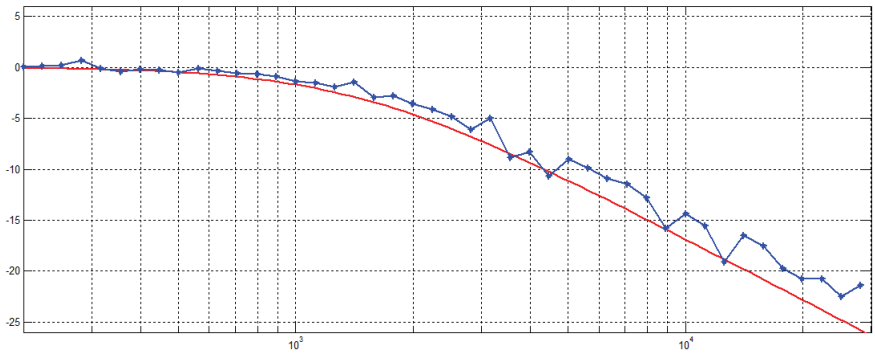

Figura 7.7: Diagrama de Bode experimental (azul) frente a la simulación (rojo).

(7.1.1) para cada frecuencia.

$$|Z| = 20 \cdot log \frac{|V_2|}{|V_1|} \tag{7.1.1}$$

Para comprobar que los resultados son precisos se comparan con una simulación generada con el programa PSPice. Como se observa en la Figura 7.7, los datos obtenidos son bastante precisos en el rango de frecuencias en que estamos trabajando. Experimentalmente se comprueba que a medida que aumenta la frecuencia se obtienen datos menos precisos, lo que es debido en gran parte a las limitaciones del convertidor analógico-digital.

Una vez obtenidos los diagramas de Bode de los patrones (véase la Figura 7.8, donde aparecen los distintos diagramas para distintas configuraciones de resistencias y condensadores) se almacena esa información para utilizarla en el clasificador donde se compararán los patrones con el diagrama de la impedancia a clasificar.

El clasificador muestra por pantalla el grado de pertenencia para cada patrón (siempre que sea mayor que el mínimo fijado, que por defecto se ha seleccionado 0.7) y finalmente a qué patrón pertenece, que sería aquel con mayor grado de pertenencia como se muestra en las figuras 7.9, 7.10 y 7.11, donde se observa que el diagrama es similar a varios patrones, pero tiene mayor afinidad con el patrón 6, y en las figuras 7.12, 7.13 y 7.14 se observan los resultados para los distintos experimentos con otros patrones (patrones 4, 5 y 8).

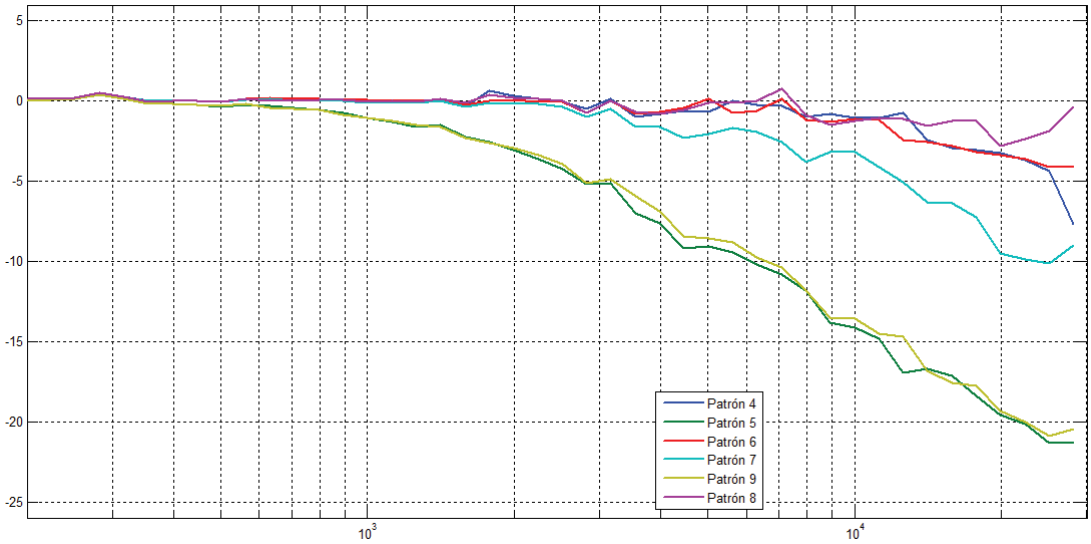

Figura 7.8: Diagrama de Bode de los diferentes experimentos de patrones de impedancia.

```
ans =

pertenece al patron 4 con un valor de pertenencia del 0.778128


ans =

pertenece al patron 6 con un valor de pertenencia del 0.863869


ans =

pertenece al patron 8 con un valor de pertenencia del 0.724758


ans =

se clasifica finalmente como el patron 6 con un valor de pertenencia del 0.863869
```

Figura 7.9: Resultado mostrado por el clasificador.

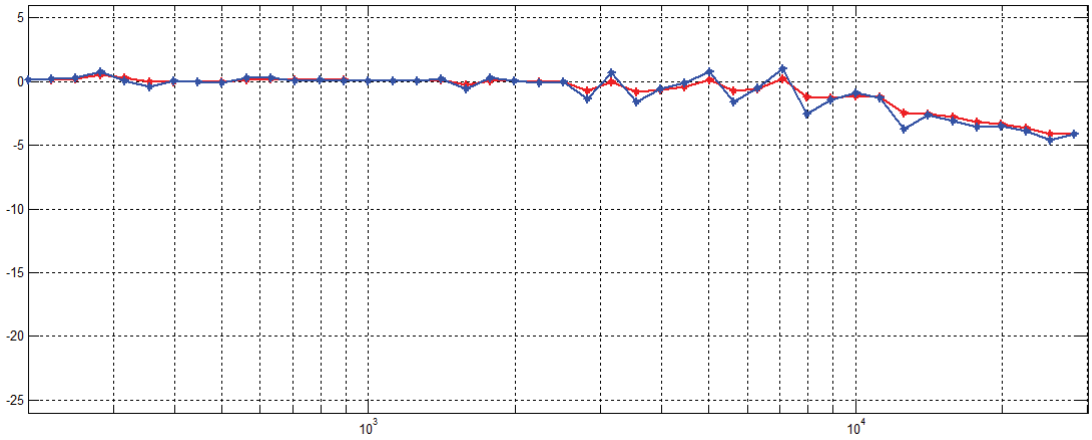

Figura 7.10: Diagrama de Bode experimental clasificado (rojo) frente al patrón 6 (azul).

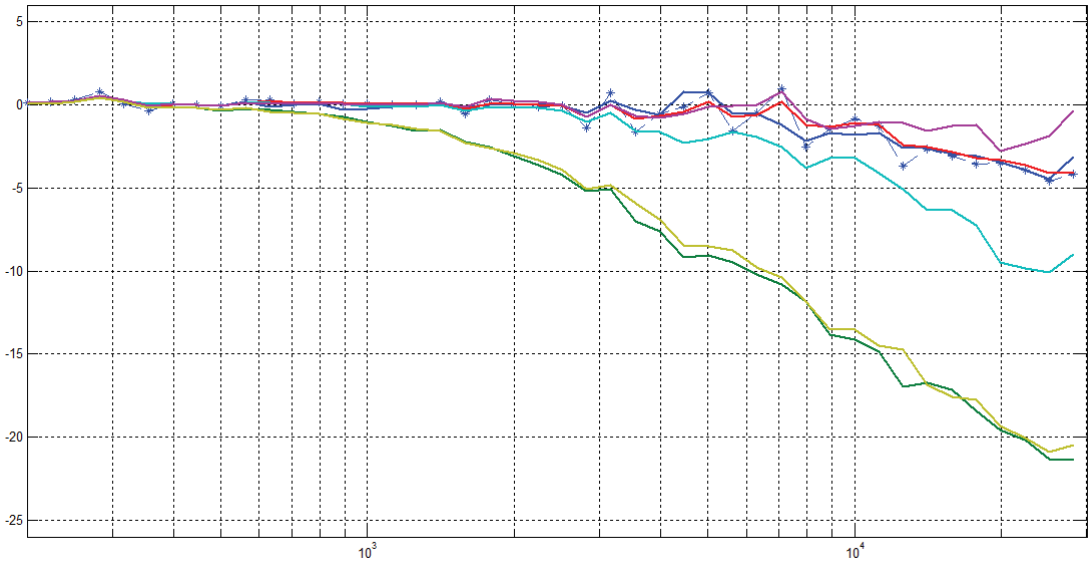

Figura 7.11: Diagrama de Bode experimental (azul *) frente a los demás patrones incluido el patrón 6.

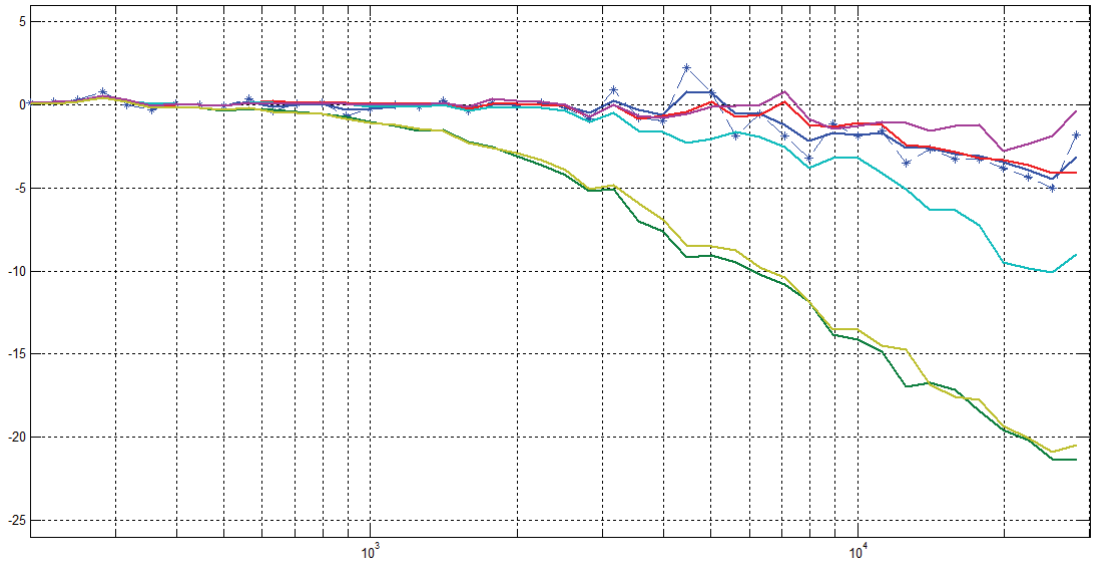

Figura 7.12: Simulación para el patrón 4 con grado de pertenencia $\mu = 0{,}839$.

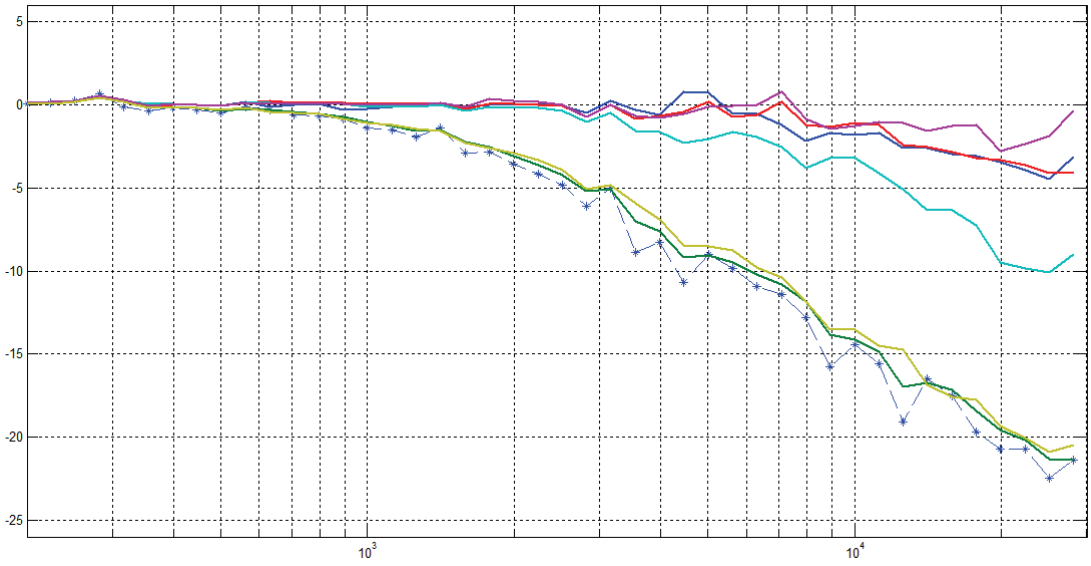

Figura 7.13: Simulación para el patrón 5 con grado de pertenencia $\mu = 0{,}74$.

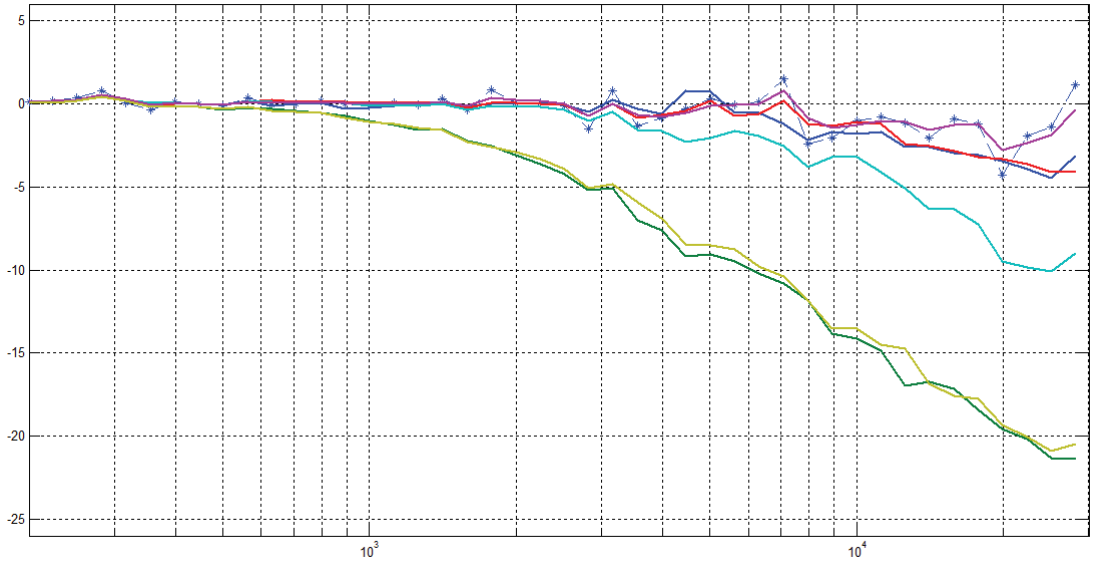

Figura 7.14: Simulación para el patrón 8 con grado de pertenencia $\mu = 0{,}83$.

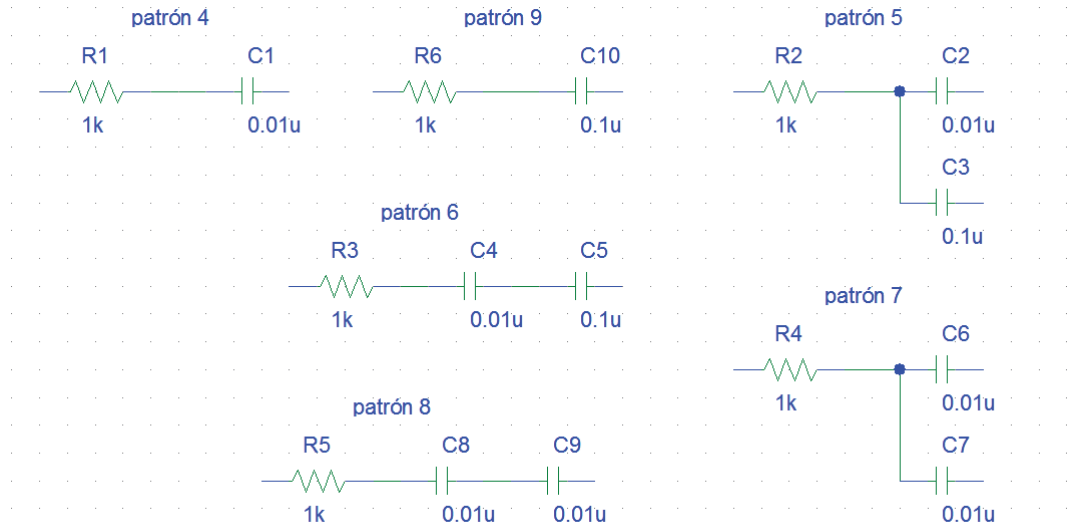

Figura 7.15: Circuitos de los patrones utilizados.

7.2. Resultados y conclusiones

Se han realizado experimentos sobre los siguientes patrones [1]:

- **Patrón 4**: $R = 1k\Omega$ en serie con $C1 = 0{,}01\mu F$
- **Patrón 5**: $R = 1k\Omega$ en serie con $C1 = 0{,}01\mu F$ y $C2 = 0{,}1\mu F$ en paralelo
- **Patrón 6**: $R = 1k\Omega$ en serie con $C1 = 0{,}01\mu F$ y $C2 = 0{,}1\mu F$ en serie
- **Patrón 7**: $R = 1k\Omega$ en serie con $C1 = 0{,}01\mu F$ y $C1 = 0{,}01\mu F$ en paralelo
- **Patrón 8**: $R = 1k\Omega$ en serie con $C1 = 0{,}01\mu F$ y $C1 = 0{,}01\mu F$ en serie
- **Patrón 9**: $R = 1k\Omega$ en serie con $C2 = 0{,}1\mu F$

En todos los experimentos se mide en el nodo que conecta la resistencia con los condensadores, (véase la Figura 7.15).

Para comprobar la eficacia del clasificador se han realizado 10 experimentos independientes para cada patrón y se han contabilizado el número

[1] no se incluyen patrones del 1 al 3 que corresponden a una resistencia, un condensador y una bobina respectivamente.

de aciertos para cada caso. Véase la tabla 7.1, de la que se pueden extraer las siguientes conclusiones:

1. Cuando se tienen patrones muy similares como lo son por ejemplo el 4 y el 6, aumenta mucho el error de clasificación. En este caso la impedancia de ambos patrones es prácticamente la misma, ya que el condensador equivalente que ve el patrón 4 es de $0{,}0091\mu F$ y el condensador correspondiente al patrón 6 es de $0{,}01\mu F$, lo que dificulta una clasificación correcta.

2. Cuando los patrones están bien diferenciados apenas hay error en la clasificación. Todos los experimentos obtienen un valor de pertenencia superior a $\mu = 0{,}7$, estando configurado el clasificador diseñado con un margen de $\pm$ 2 dB correspondientes a los parámetros a y b del clasificador triangular visto en 5.3.1.

Patrón	1	2	3	4	5	6	7	8	9	10	total aciertos
4	0	0	0	0	1	0	0	0	0	1	20 %
5	1	1	1	1	1	1	1	1	1	1	100 %
6	1	1	1	1	1	1	1	0	1	1	90 %
7	1	1	1	1	1	1	1	1	1	1	100 %
8	1	1	1	1	1	1	1	1	1	1	100 %
9	1	1	1	1	1	1	1	0	1	1	90 %

Cuadro 7.1: Resultados de los 10 experimentos sobre los circuitos patrones.

Por último vamos a comprobar la veracidad del clasificador cuando se miden circuitos distintos a los circuitos patrón, con valores de impedancias desconocidos y distintos a las impedancias patrón (figuras 7.17, 7.18, 7.19, 7.20, 7.21).

Como puede verse el clasificador continúa clasificando las impedancias experimentales correctamente entre los patrones conocidos aunque el valor de pertenencia μ es mucho menor que en el caso anterior, incluso menor al valor límite que se ha establecido previamente ($\mu = 0{,}7$). A continuación se pueden consultar los resultados de estos experimentos, y los circuitos utilizados.

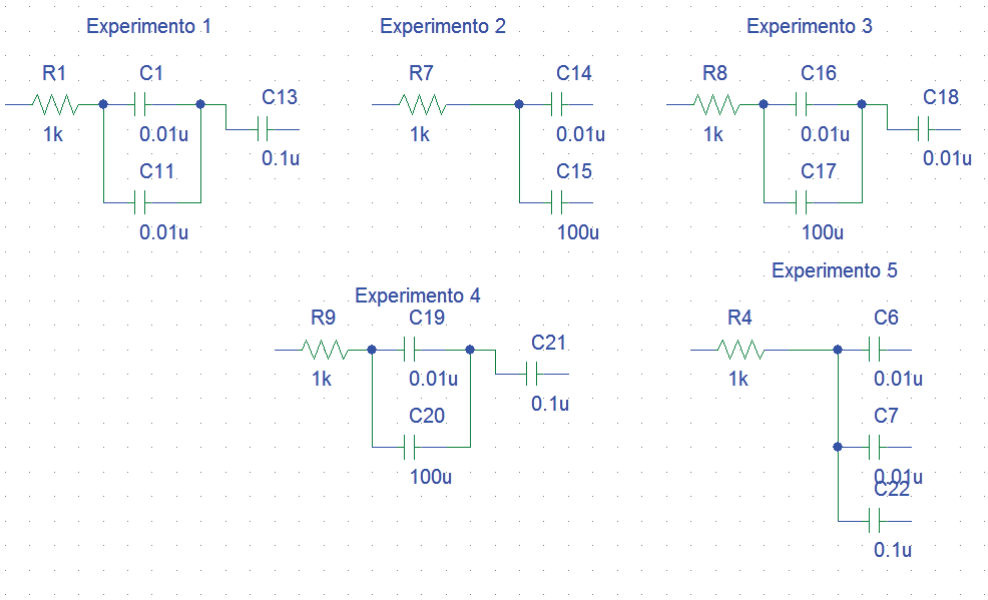

Figura 7.16: Circuitos experimentales utilizados.

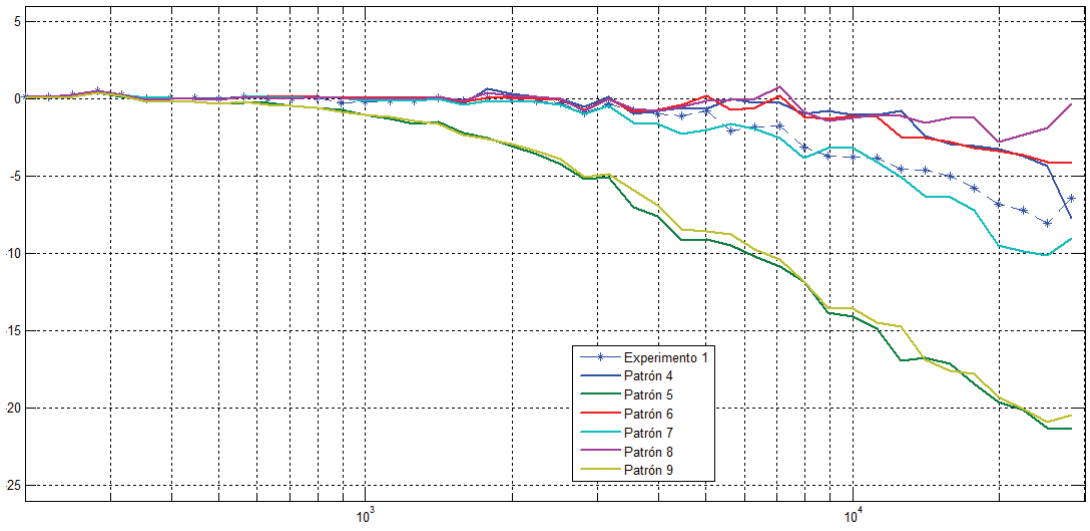

Figura 7.17: Experimento 1 se clasifica con el patrón 7 con $\mu = 0{,}74$.

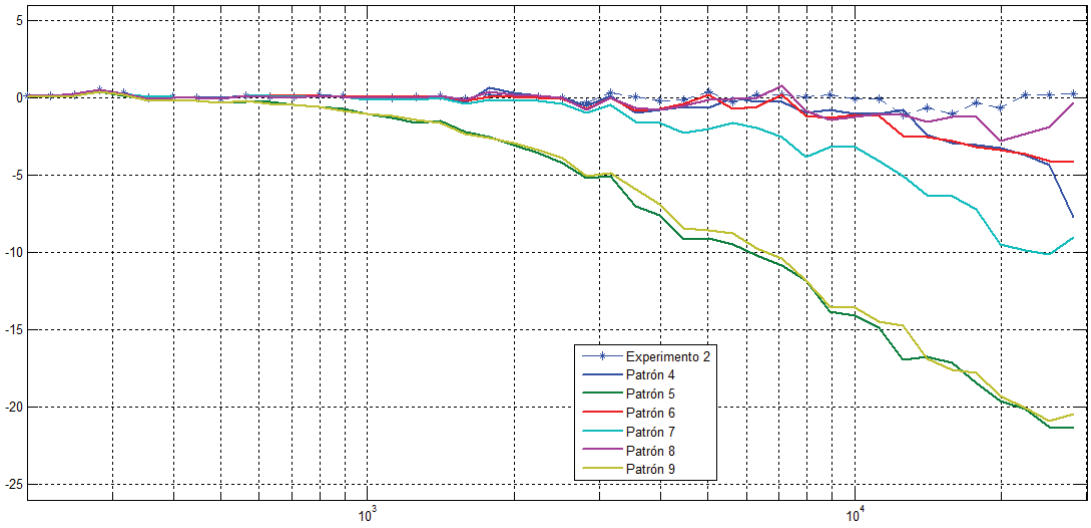

Figura 7.18: Experimento 2 se clasifica con el patrón 8 con $\mu = 0{,}79$.

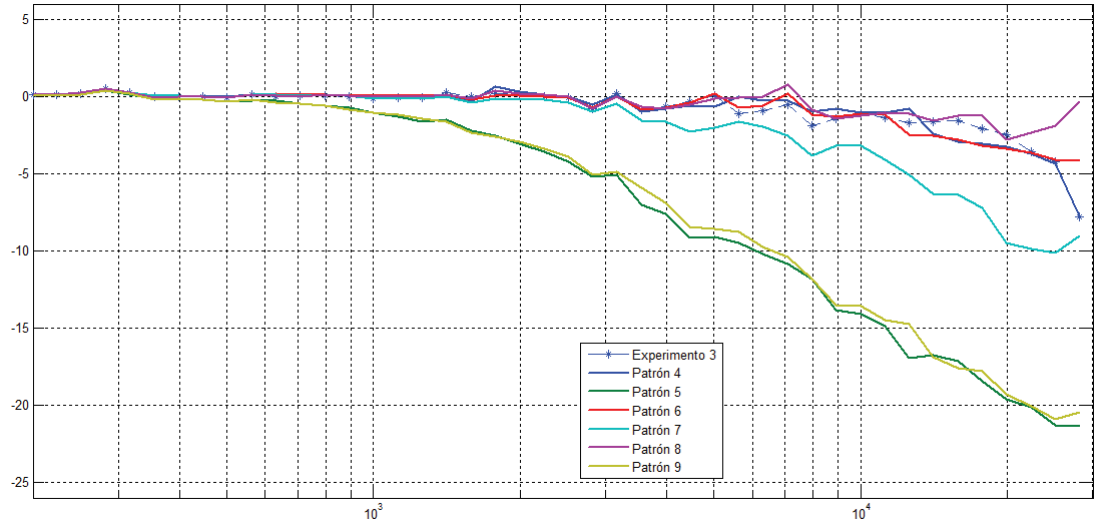

Figura 7.19: Experimento 3 se clasifica con el patrón 6 con $\mu = 0{,}87$.

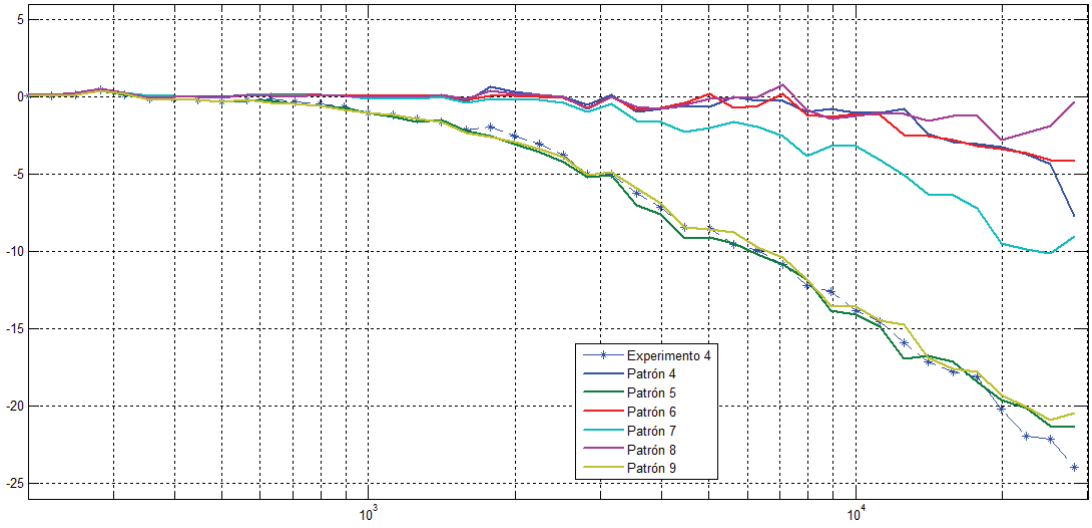

Figura 7.20: Experimento 4 se clasifica con el patrón 9 con $\mu = 0{,}83$.

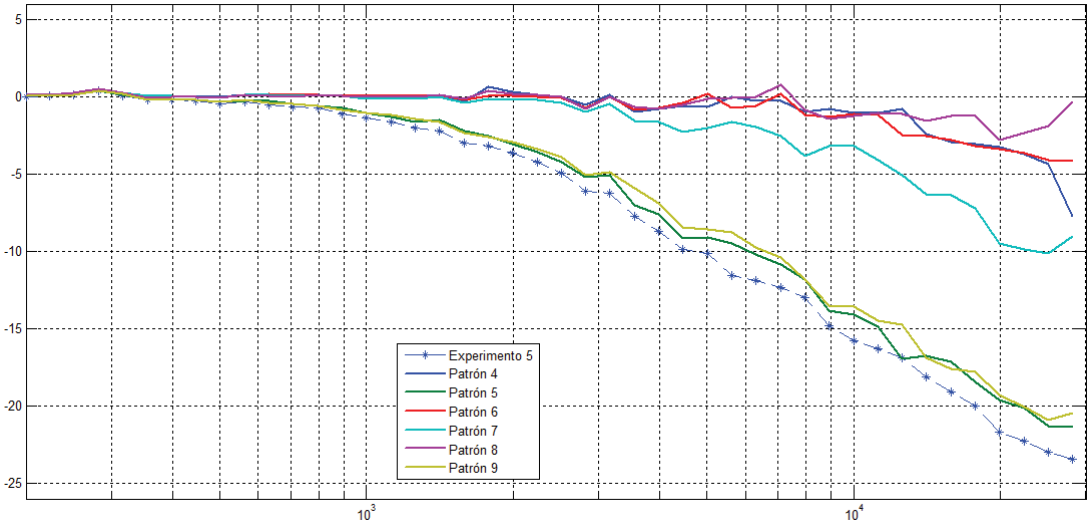

Figura 7.21: Experimento 5 se clasifica con el patrón 5 con $\mu = 0{,}59$.

Capítulo 8

Medición de Bioimpedancia

En este capítulo se aplica los conocimientos adquiridos durante el proyecto a la clasificación de bioimpedancias. Para ello se han colocado las bornas de los pin de entrada y tierra de Arduino al dedo del sujeto de pruebas, tal y como puede verse en la Figura 8.1, y se han realizado experimentos en contacto con distintos objetos, véase la Figura 8.2.

8.1. Patrones

Los resultados obtenidos muestran la gran variabilidad de la bioimpedancia según los materiales u objetos tocados, como se muestra en la Figura 8.3, donde se observan diferentes respuestas para los 16 experimentos [1] distintos de la lista siguiente:

- tocar el dedo pulgar (13)
- tocar la parte metálica del PC (14)
- tocar plástico (15)
- tocar jarrón de cristal (16)
- tocar jarrón lleno de agua (17)
- agarrar bote de metal (18)
- agarrar un boligráfico (19)
- agarrar jarrón lleno de agua (20)
- tocar bote de metal (21)
- tocar y agarrar plaquita metálica (22 y 23)

[1] se han descartado los patrones 1-12 debido a la colocación de la sonda.

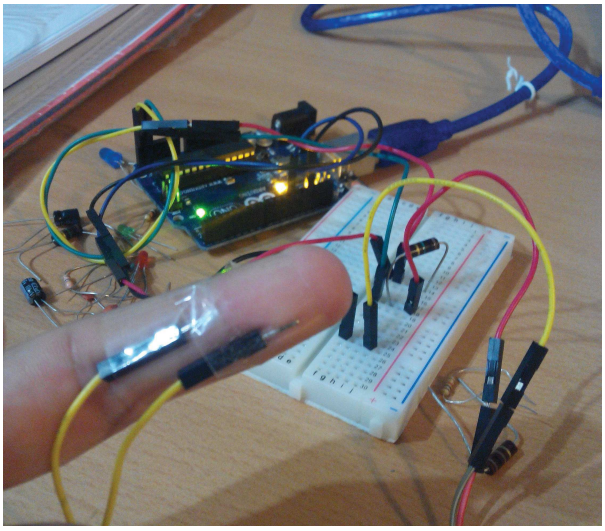

Figura 8.1: Montaje del sensor de bioimpedancia.

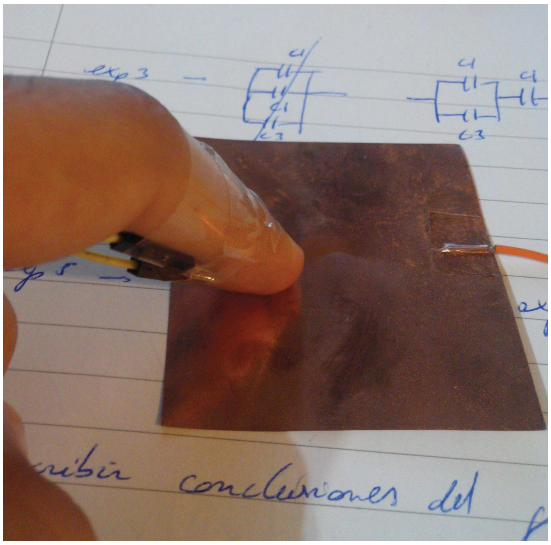

Figura 8.2: Experimento tocando plaquita metálica de prueba.

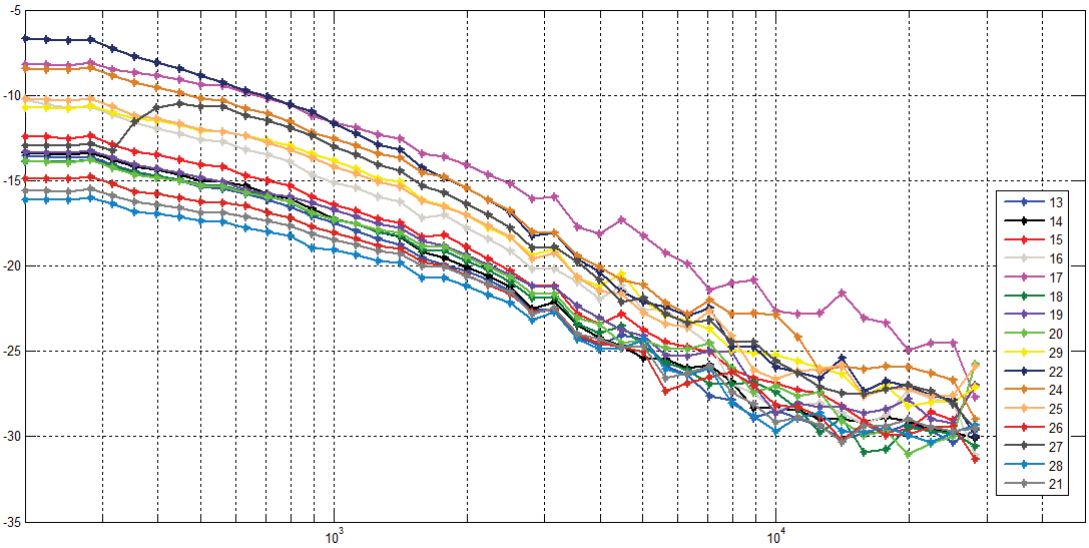

Figura 8.3: Diferentes valores de bioimpedancia.

- no tocar nada (24)
- tocar el dedo índice de la otra mano (25)
- mojar el dedo en el agua con y sin tocar con el resto de la mano el jarrón (26 y 27)
- coger y tocar botella de plástico con agua (28 y 29)

8.2. Clasificador de bioimpedancia

Ahora que hemos conseguido hasta 16 patrones distintos se realizan experimentos para probar el funcionamiento del clasificador de impedancias anterior aplicado a la bioimpedancia. Los resultados obtenidos no son los esperados debido probablemente a que estos experimentos se han realizado en unas condiciones distintas a las de la adquisición de los patrones (distinto día, colocación de la sonda distinta, contacto sonda-piel, humedad al tocar el agua, colocación del resto del cuerpo distinta, distinta ropa y accesorios como un reloj...), lo que afecta mucho a la correcta repetibilidad de los experimentos, siendo muy dificil clasificar correctamente usando los patrones del día anterior, tal y como se muestran en la Figura 8.4.

En vista de lo anterior, se decide realizar de nuevo la adquisición de 6 pa-

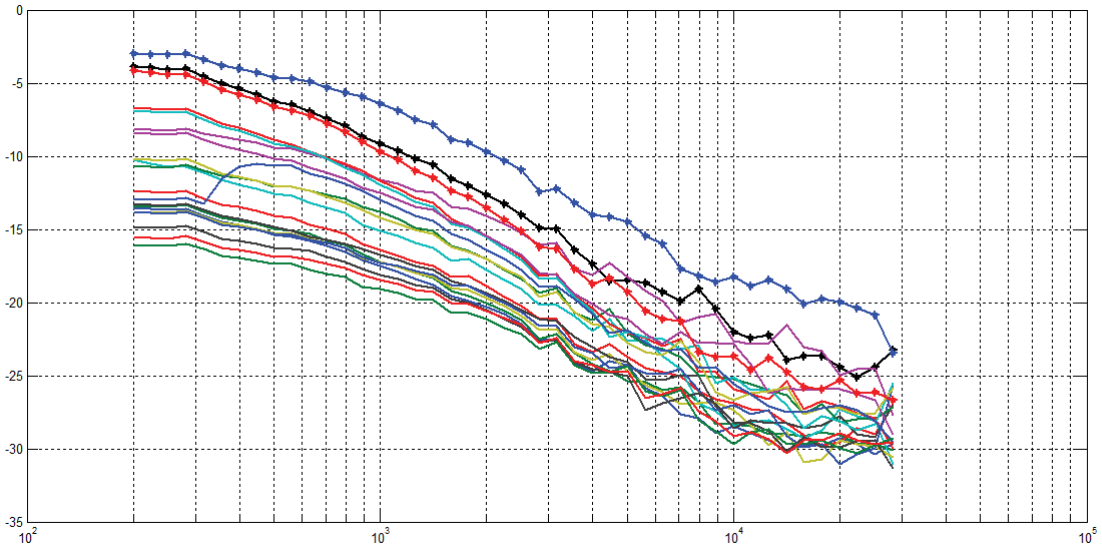

Figura 8.4: Se observa que los diagramas experimentales (*) no coinciden con los patrones.

trones [2], para repetir los experimentos intentando mantener las condiciones no medidas lo más constante posible para que no afecten al resultado. En esta ocasión los resultados mostrados en la tabla 8.1, son más favorables logrando clasificar correctamente varios patrones. Nótese que en algunos casos se obtienen resultados muy similares al realizar la medida, lo que dificulta diferenciar correctamente un patrón de otro (caso del patrón 5), véase figuras 8.5, 8.6, 8.7.

Patrón	1	2	3	4	5	total aciertos
1	0	1	1	1	0	60 %
2	0	1	1	1	1	80 %
3	0	1	1	1	0	60 %
4	1	1	1	1	0	60 %
5	0	0	0	0	0	0 %
6	1	1	1	1	1	100 %

Cuadro 8.1: Resultados de los 5 experimentos sobre los nuevos patrones.

[2] entiendase por patrón la curva obtenida como referencia, y se demuestra que al hacer varios experimentos el resultado se ajusta a los patrones de referencia

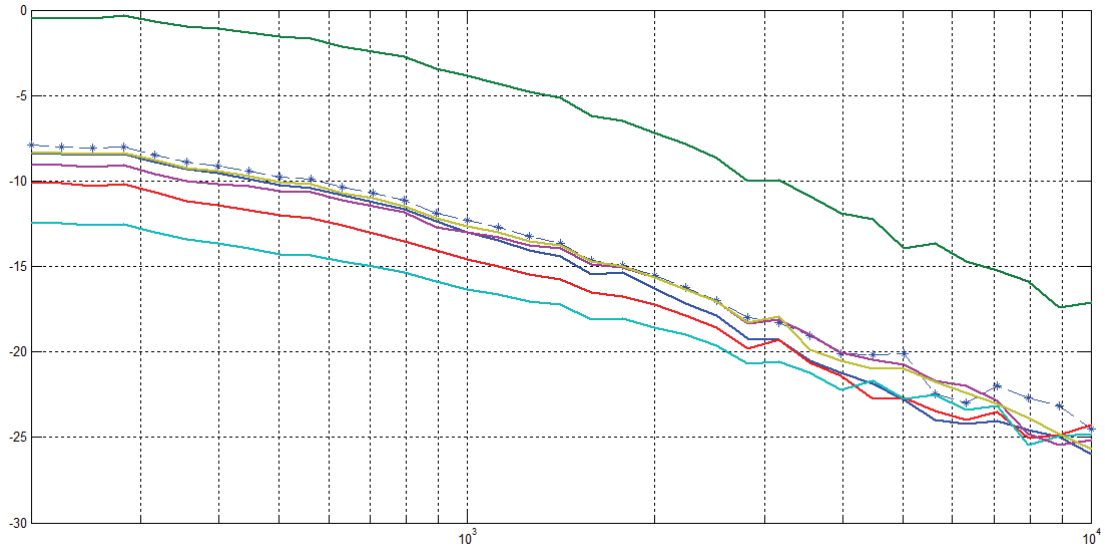

Figura 8.5: Se observa que los patrones 5 y 6 son muy similares, por tanto realiza una clasificación errónea

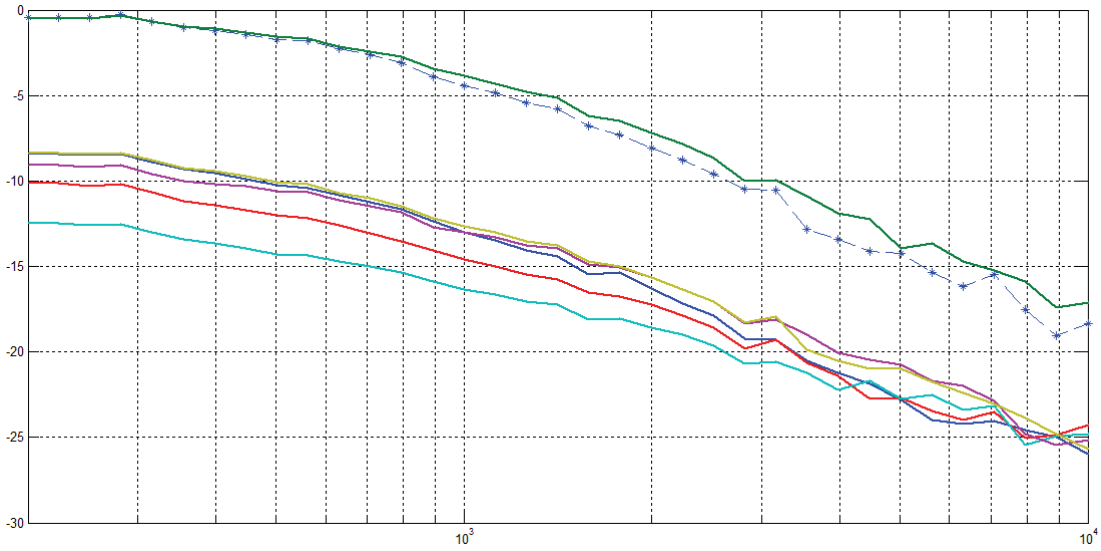

Figura 8.6: Se observa que en este caso clasifica correctamente con el patrón correcto

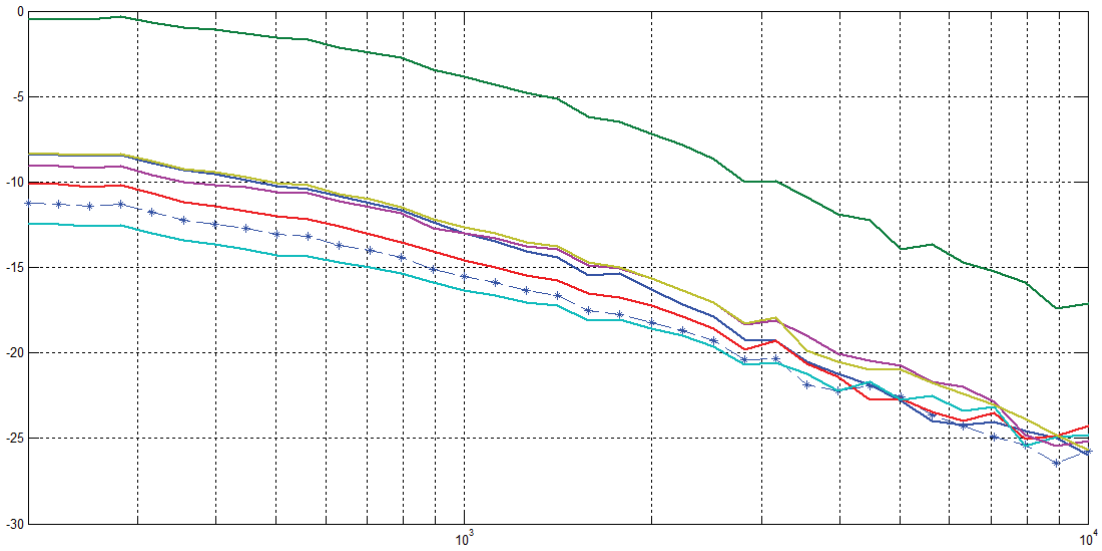

Figura 8.7: Se observa que la bioimpedancia medida pertenece al patrón 3 con $\mu = 0{,}72$, aunque se encuentra entre el 3 y el 4

8.3. Adquisición rápida de datos y correcta clasificación

Para terminar se ha realizado un experimento para mejorar la velocidad de adquisición de datos, para lo que se han tomado las siguientes medidas:

- disminuir el número de puntos de 44 a 20 valores de frecuencia
- disminuir el rango de frecuencia leído, de $32kHz$ a $20KHz$ de frecuencia máxima
- disminuir el número de muestras para cada frecuencia, de 320 a 68

Con estos cambios se consigue disminuir el tiempo entre experimentos. De 30 segundos por experimento se ha pasado a 8 segundos clasificando correctamente la bioimpedancia. En las figuras 8.8 y 8.9 por ejemplo, se clasifica correctamente entre dos casos, tocar o no tocar un jarrón de cristal con agua, usando este nuevo método más rápido.

En la Figura 8.10 se observa claramente la distribución en torno a un patrón u otro de los 5 experimentos realizados y en en la tabla 8.2 se encuentran los resultados.

La conclusión de este capítulo es que la bioimpedancia varía mucho en un pequeño rango y depende de factores experimentales difíciles de medir y de mantener constantes. Con todo, si se realizan suficientes experimentos

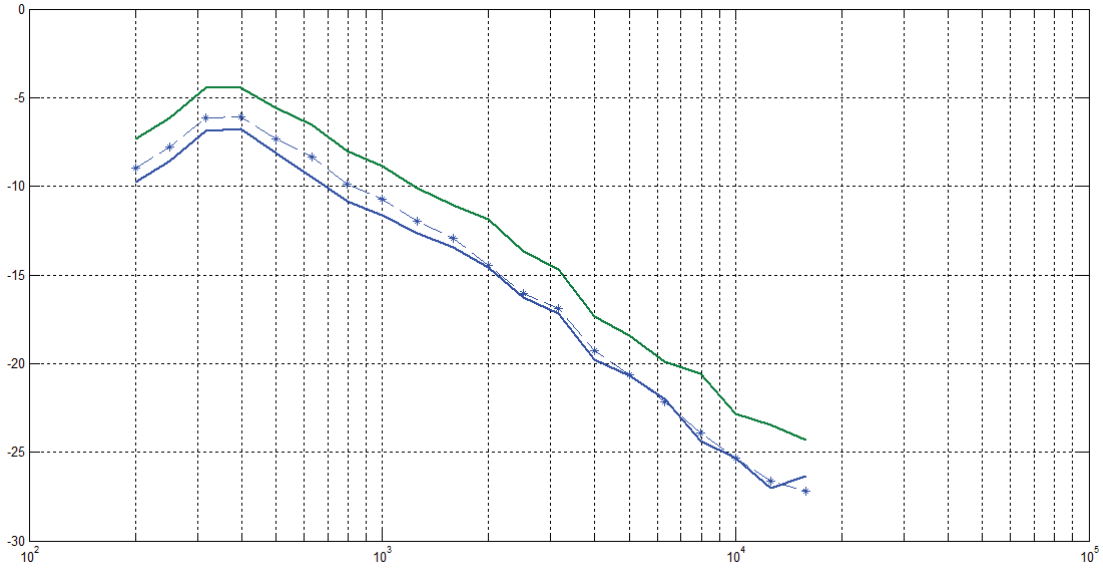

Figura 8.8: Se observa que la bioimpedancia medida pertenece al patrón 1 (no tocar) con grado de pertenencia $\mu = 0{,}81$.

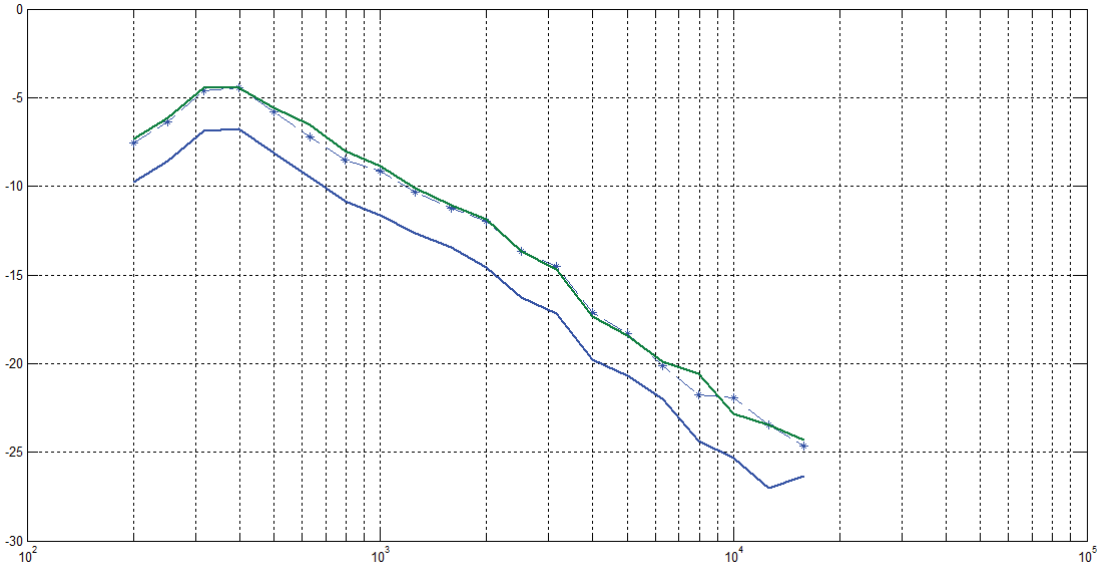

Figura 8.9: Se observa que la bioimpedancia medida pertenece al patrón 2 (tocar cristal) con grado de pertenencia $\mu = 0{,}89$.

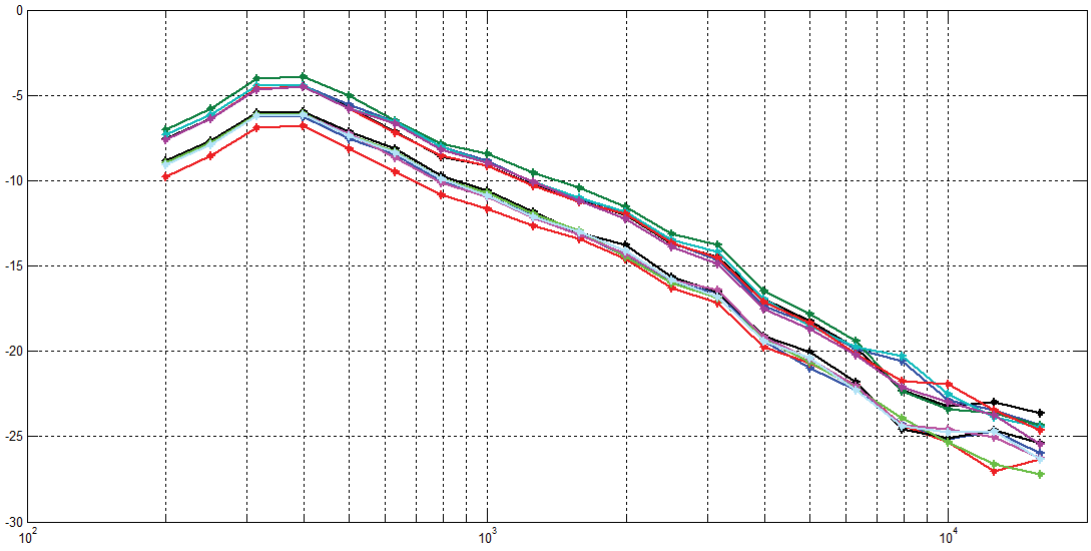

Figura 8.10: Se observa que la bioimpedancia medida se distribuye en torno a dos patrones.

Patrón	1	2	3	4	5	total aciertos
Sin tocar	1	1	1	1	1	100 %
Tocar cristal	1	1	1	1	1	100 %

Cuadro 8.2: Resultados de los 5 experimentos sobre los dos patrones.

se ajusta a unos patrones fijos y comunes, lo que permite clasificarla correctamente y diferenciar entre tocar agua, agarrar cristal, tocar un dedo, etc.

Capítulo 9

Conclusiones

Una vez realizados los experimentos y comprobados los resultados, se concluye que la plataforma Arduino es apta para la adquisición de datos a bajo coste con una precisión aceptable (10 bits de resolución), pero con limitaciones como la frecuencia de reloj máxima (de $16MHz$), la baja memoria interna y la existencia de un único convertidor analógico-digital, que además es lento en comparación con cualquier tarjeta de adquisición de datos, lo que limita el rango de frecuencias alcanzable y el rango de impedancias que se pueden clasificar.

Una solución a estos problema es añadir un circuito de acondicionamiento de señal, que coloque la impedancia en el rango de lectura, limitando la intensidad que se le aporta. Algo tan simple como una resistencia variable, permitiría observar un mayor rango de capacidades.

Se ha comprobado también la gran versatilidad que añade un clasificador basado en Lógica Borrosa al problema de clasificación de impedancias que nos atañe, lo sencillo que es de implementar y cómo trabaja cuando existe incertidumbre entre los distintos patrones.

Durante la realización del proyecto nos hemos encontrado con una serie de dificultades que se han ido solucionando o corrigiendo en mayor o menor medida, las más acusadas son las correspondientes a la plataforma Arduino por las limitaciones técnicas comentadas anteriormente.

A pesar del simple circuito que se ha usado en estos experimentos, el circuito de acondicionamiento de señal afecta en parte a los resultados obtenidos, por tanto es un detalle importante a la hora de realizar los experimentos, en proyectos futuros sería importante reducir el impacto del circuito de acondicionamiento al mínimo, ya que afectará esa configuración a experimentos posteriores.

Es necesario resaltar la dificultad de una correcta comunicación entre Arduino y Matlab que impide poder procesar los datos en tiempo real desde

Matlab, y obliga a recoger primero todos los datos, debido a que el sistema de comunicación no está optimizado como ocurriría con una tarjeta de adquisición de datos y su correspondiente software.

Finalmente, como se ha comentado en capítulos anteriores, las condiciones del experimento afectan mucho al resultado al medir la bioimpedancia, es difícil volver a repetir un experimento en las mismas condiciones.

En un futuro sería muy recomendable solucionar los problemas de acondicionamiento de señal y diseñar una sonda correcta para mejorar las condiciones al realizar los experimentos y obtener resultados que se ajusten a la realidad.

Además de las mejoras de circuito sería muy interesante realizar todo el procesamiento en la plataforma Arduino (quizás utilizando otro método de clasificación con menos requerimiento computacional, como por ejemplo el Análisis de Componentes Principales (PCA)), evitando así problemas de comunicación y convirtiendo el clasificador en una plataforma portátil.

Por último comentar que este proyecto se comenzó con la intención de aplicar esa clasificación al campo de la domótica (por ejemplo activar algún elemento de un sistema domótico al diferenciar entre distintos gestos, o materiales), por tanto queda como idea futura implementar algún método de conexión de la plataforma Arduino con alguna interfaz domótica y actuar sobre ella mediante acciones dependientes de la bioimpedancia.

Apéndices

Apéndice A

Demostración de la Transformada Discreta de Fourier (DFT)

Como se ha comentado previamente, usaremos la DFT en este proyecto debido a que tenemos señales muestreadas en el tiempo, en este anexo se amplía la información relativa a la DFT. Consideremos una secuencia $x[n]$ de longitud finita, $0 \leq n \leq N-1$. Su transformada Z vendrá dada por:

$$X(z) = \sum_{n=0}^{N-1} x[n] z^{-n} \tag{A.0.1}$$

Donde $X(z)$ es una función de variable compleja z. Puesto que $x[n]$ es una secuencia de longitud finita N y causal, la región de convergencia de $X(z)$ es todo el plano Z excepto el origen. Los N puntos de la Transformada de Fourier Discreta $X[k]$ de la secuencia $x[n]$ considerada se definen como muestras frecuenciales $X(z_k)$, que se obtienen mediante el muestreo de $X(z)$ en N puntos igualmente espaciados z_k del círculo unidad,$z = e^{jw}$, dados por:

$$z_k = e^{j\frac{2\pi}{N}k} \quad 0 \leq k \leq N-1$$

$$X[k] = X(z) \mid_{z=e^{j\frac{2\pi}{N}k}} = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}, \quad 0 \leq k \leq N-1. \tag{A.0.2}$$

Haciendo un cambio de notación en A.0.2 del tipo,

$$W_N = e^{-j\frac{2\pi}{N}},$$

la expresión para el cálculo de la DFT puede ser reescrita de la siguiente manera

$$X[k] = \sum_{n=0}^{N-1} x[n] W_n^{kn}, \quad 0 \leq k \leq N-1. \tag{A.0.3}$$

Apéndice B

Demostración de la Transformada Rápida de Fourier (FFT)

En este apartado se explica en detalle cómo funciona el algoritmo de la FFT en el que se basa la función de MATLAB®, se trata del algoritmo de Cooley-Tukey (FFT-CT) comentado previamente [9] :

Dada una DFT de tamaño M, la descompone en dos DFTs entrelazadas de tamaño M/2 en cada paso iterativo. El primer paso calcula la DFT de las dos secuencias alternas:

$$x_{2m} = \{x_0, x_2, ..., x_{m-2}\}$$

$$x_{2m+1} = \{x_1, x_3, ..., x_{m-1}\}$$

Y a continuación se combinan esos dos resultados para producir la DFT completa. Este procedimiento puede iterarse para cada subsecuencia, siempre generando un par adicional de secuencias hasta que se llega a la secuencia irreducible que tiene solución trivial. Así, la ecuación de transformada DFT quedaría descompuesta en el primer paso como la suma de una DFT para la secuencia par y otra DFT para la secuencia impar:

$$X(k) = \sum_{n=0}^{(M/2)-1} x(2n)e^{-\frac{2\pi i}{M/2}mk} + \sum_{n=0}^{(M/2)-1} x(2n+1)e^{-\frac{2\pi i}{M/2}mk} = \quad \text{(B.0.1)}$$

$$= E_k + e^{-\frac{2\pi i}{M/2}k}O_k$$

Donde E_k es la nueva DFT generada desde la secuencia de datos pares y O_k es la nueva DFT generada desde la secuencia de datos impares. Cada nueva DFT es una secuencia con M/2 elementos, de modo que hay que

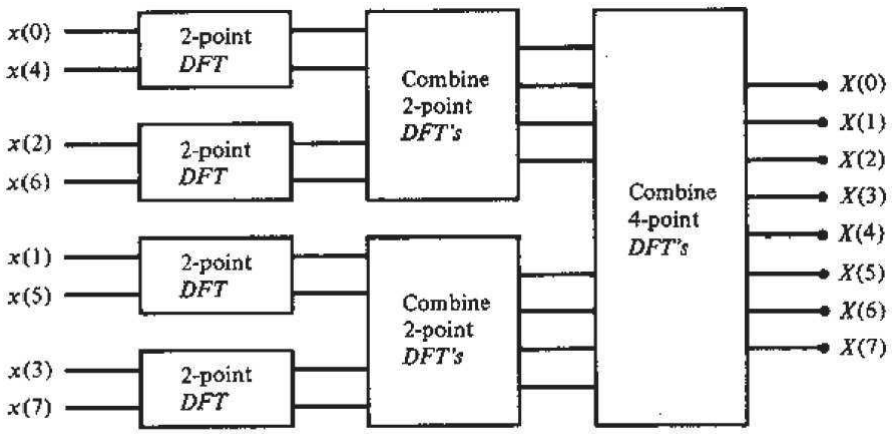

Figura B.1: Cómputo de una DFT de 8 puntos en tres etapas

calcular M/2 sumas. Gracias a las propiedades de periodicidad del factor de fase W_k^M , los elementos de la secuencia de $X(kF)$ entre $M/2 \leq k < M$son idénticos a los elementos de la secuencia $X(kF)$ entre $0 \leq k < M/2$, es decir, se tienen las relaciones:

$$E_{k+M/2} = E_k$$

$$O_{k+n/2} = O_k$$

El factor de fase tiene además la propiedad de simetría, cambiando el signo de los términos $O_{k+M/2}$ de modo que toda la DFT se puede calcular como:

$$X(k) = \begin{cases} E_k + e^{-\frac{2\pi i}{M/2}k} O_k & sik < M/2 \\ E_{k-M/2} + e^{-\frac{2\pi i}{M/2}k-M/2} O_{k-N/2} & sik \geq M/2 \end{cases} \tag{B.0.2}$$

Que expresa la secuencia DFT $X(kF)$ de longitud N en términos de dos DFTs de tamaño M/2. Aplicando ésta fórmula iterativamente a cada nueva pareja de DFTs con la mitad de elementos se termina calculando una secuencia DFT compuesta por sólo dos términos $X(kF)$, correspondiente a una secuencia de datos de dos elementos.

Apéndice C

Código Arduino comentado

```
//Programa para la adquisición de datos,
//generando un PWM con corrección de fase y de frecuencia
   #include <math.h>

//constantes para seleccionar el preescalado
// para calcular la frecuencia del ADC

const unsigned char PS_2 = (1 << ADPS0);
const unsigned char PS_4 = (1 << ADPS1);
const unsigned char PS_8 = (1 << ADPS1) | (1 << ADPS0);
const unsigned char PS_16 = (1 << ADPS2);
const unsigned char PS_32 = (1 << ADPS2) | (1 << ADPS0);
const unsigned char PS_64 = (1 << ADPS2) | (1 << ADPS1);
const unsigned char PS_128 = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);



#define N 320                //dimensión de los vectores de datos

int vin[N];                 //vector de valores de la señal de entrada
int vout[N];               //vector de valores de la señal de salida
int vx;

unsigned long F;                      //variable frecuencia
unsigned long B;
unsigned long A;
unsigned long C;
unsigned long D;
double F1;
```

```
float F2;
float i;

void setup()
{



  pinMode(3,OUTPUT);
  pinMode(11,OUTPUT);  //pin PWM

  pinMode(A5,INPUT);   //pin lectura vin
  pinMode(A4,INPUT);   //pin lectura vout

  pinMode(A0,INPUT);

 //configuracion para el PWM

  TCCR2A = _BV(COM2A0) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
  TCCR2B = _BV(WGM22) |  _BV(CS22)  ;



  Serial.begin(115200);      //velocidad de transmisión del puerto serie

  for(int i=0;i<N;i++)     //bucle for de inicialización de los vectores
    {
      vin[i]=0;

      vout[i]=0;
    }




}

void loop()
{

//bucle for de selección de frecuencias, en cada iteración aumenta
//la frecuencia del PWM de forma logaritmica

  for( unsigned long j=2300;j<=4500;j=j+50)
```

```
    {
         //operacion para el calculo de la frecuencia correcta
//+0.5 para que al convertir a entero redondee correctamente

 i=(float)j/1000;
   F1=pow(10,i)+0.5000;
   F2=(float)F1;
   F=(unsigned long)F2;


//para cada rango de F aplicamos distintos PS y ADPS

  if (F<=130)
  {


    // seleccionamos el PS=1024 del PWM

    TCCR2B = _BV(WGM22) | _BV(CS22) | _BV(CS21) | _BV(CS20) ;


//fórmula que calcula el ICR2 que hay que aplicar para
//obtener la frecuencia F

   C=1024*F*2;
   B=(16000000/C);
  A=B-1;
  D=A/2;

  OCR2A = A;
  OCR2B = D;
  vx= analogRead(A0);

  }

  if (F>130 && F<=250)
  {



    TCCR2B = _BV(WGM22) | _BV(CS22) | _BV(CS21) ; //PS=256
```

```
 C=256*F*2;
 B=(16000000/C);
A=B-1;
D=A/2;

OCR2A = A;
OCR2B = D;
vx= analogRead(A0);

}

  if (F>250 && F<=500)
{



  TCCR2B = _BV(WGM22) | _BV(CS22) | _BV(CS20) ; //PS=128

 C=128*F*2;
 B=(16000000/C);
A=B-1;
D=A/2;

OCR2A = A;
OCR2B = D;
vx= analogRead(A0);

}

if (F>500 && F<=1000)
{



  TCCR2B = _BV(WGM22) | _BV(CS22) ; //PS=64

 C=64*F*2;
 B=(16000000/C);
A=B-1;
D=A/2;

OCR2A = A;
OCR2B = D;
```

```
  vx= analogRead(A0);

  }

  if (F>1000 && F<=2500)
  {


    TCCR2B = _BV(WGM22) | _BV(CS21) | _BV(CS20) ; //PS=32

   C=32*F*2;
   B=(16000000/C);
  A=B-1;
  D=A/2;

  OCR2A = A;
  OCR2B = D;
  vx= analogRead(A0);

  }

  if (F>2500 && F<=4000)
  {

     ADCSRA &= ~PS_128;
// elimina el bit puesto por la libreria Arduino
 // seleccionamos uno de los de arriba
  // PS_16, PS_32, PS_64 or PS_128

  ADCSRA |= PS_64;    // set our own prescaler to 64


     TCCR2B = _BV(WGM22) | _BV(CS21) | _BV(CS20) ; //PS=32

   C=32*F*2;
   B=(16000000/C);
  A=B-1;
  D=A/2;

  OCR2A = A;
  OCR2B = D;
  vx= analogRead(A0);

  }
```

```
if (F>4000 && F<=5000)
{



  TCCR2B = _BV(WGM22) | _BV(CS21)  ; //PS=8

 C=8*F*2;
 B=(16000000/C);
A=B-1;
D=A/2;

OCR2A = A;
OCR2B = D;
vx= analogRead(A0);

}

if (F>5000 && F<=8000)
{

  ADCSRA &= ~PS_64;

// PS_16, PS_32, PS_64 or PS_128

ADCSRA |= PS_32;    // set our own prescaler to 32


  TCCR2B = _BV(WGM22) | _BV(CS21)  ; //PS=8

 C=8*F*2;
 B=(16000000/C);
A=B-1;
D=A/2;

OCR2A = A;
OCR2B = D;
vx= analogRead(A0);

}
```

```
if (F>8000 && F<=18000)
{

   ADCSRA &= ~PS_32;

// PS_16, PS_32, PS_64 or PS_128

ADCSRA |= PS_16;    // set our own prescaler to 16


  TCCR2B = _BV(WGM22) | _BV(CS21)  ; //PS=8

 C=8*F*2;
 B=(16000000/C);
A=B-1;
D=A/2;

OCR2A = A;
OCR2B = D;
vx= analogRead(A0);

}

if (F>18000 && F<=25000)
{




  TCCR2B = _BV(WGM22) | _BV(CS21)  ; //PS=8

 C=8*F*2;
 B=(16000000/C);
A=B-1;
D=A/2;

OCR2A = A;
OCR2B = D;
vx= analogRead(A0);

}
```

```
  if (F>25000 && F<=32000)
  {



    TCCR2B = _BV(WGM22) | _BV(CS21)  ; //PS=8

   C=8*F*2;
   B=(16000000/C);
  A=B-1;
  D=A/2;

  OCR2A = A;
  OCR2B = D;
  vx= analogRead(A0);

  }




  analogRead(A5);

  for( int i=0;i<=N;i++)        //bucle for de lectura
    {

      vin[i]= analogRead(A5);
//leo la señal de entrada por el pin analógico A5
//la almaceno en el vector de daos que envio
//por el puerto serie


    }


  delay(100);



 ////////**************************////////////////


  analogRead(A0);
```

```
  analogRead(A4);
  for( int i=0;i<=N;i++)        //bucle for de lectura
    {

      vout[i]= analogRead(A4);

//leo la señal de entrada por el pin analógico A4


    }


  //apaga PWMgen
  TCCR2A = _BV(COM2A0) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
  TCCR2B = _BV(WGM22) ;

 delay(100);

  for (int i=0; i <= N; i++)
//bucle de escritura,
//envia por puerto serie una linea con el
//valor "i" de cada vector separado por ","
    {




    Serial.print(F);
    Serial.print(",");
    Serial.print(vin[i]);
    Serial.print(",");
    Serial.println(vout[i]);

    }


    }


  delay(100);



  }
```

Apéndice D

Código Matlab comentado

```
%%primero introducimos los datos del experimento

clear all
        clc
close('all')
format long g

%abrir archivos de texto

[archivo,ruta]=uigetfile('*.txt','ABRIR ARCHIVO');
if archivo==0
    return;
else
fid =fopen([ruta archivo],'r');
A=textscan(fid,'%f,%f,%f');
A=cell2mat(A);
fclose(fid);

end
Fard=A(:,1);        %frecuencia a la que exigimos que trabaje
vin=A(:,2)*5/1024;  %valor de tension que aplicamos
vout=A(:,3)*5/1024; %valor de tension que leemos

%elimina outliers o errores de lectura
z=length(vin);

for i=2:(z)
   if(vin(i)>5 || vin(i)<0)
```

```
        vin(i)=vin(i-1);
    end
    if(vout(i)>5 || vout(i)<0)
        vout(i)=vout(i-1);
    end

end



%ahora pasamos al calculo del bode, centrarnos solo en la ganancia

%P es la frecuencia tope a la que trabajamos en Hz
P=31623;

%parámetros de diseño para el correcto calculo del bode
R=1000;

L1=316;
L2=15;
Fint=4000;

per=0.05;

%llama a la funcion diagrama de bode

[bode,bode1,s1]=Bode10(Fard,vin,vout,P,R,L1,L2,Fint,per);

semilogx(bode(:,1),bode(:,2),'*-','LineWidth',2)
    axis([200 30000 -26 6])
        grid


figure()

%llama a la funcion clasificador
[xp, yp]=clasif(bode1);


%fin del programa principal

%funciones utilizadas
```

```
function [bode,bode1,s1] = Bode10(Fard,vin,vout,P,R,L1,L2,Fint,per )


i=1;
fcont=200;  %frecuencia a la que hacemos el calculo
m=1;
% P es la frecuencia tope de muestras
 %j es la frecuencia inicial 10^j
j=2.3;
%L es un parámetro para variar el numero de datos
%que se envian a la funcion FFT, se obtiene mayor
%precision con L grandes para baja frecuencia y
%L pequeña para baja frecuencia
L=L1;

while Fard(i)<P


    if Fard(i)==fcont

      if fcont>Fint
          L=L2;
      end
%

    %rango y datos a usar para esta F
    %la i indica que estamos dentro del rango para esa frecuencia Fard
    %la h marca el final del rango para que usemos el mismo rango
%que el de la frecuencia elegida
    iran=i+1;
    fran=iran+L;

    s1=vin(iran:fran);
    s2=vout(iran:fran);

    %calculo de magnitud en dB
    %calculamos la ganancia de vin y vout para construir el bode

    X1=fft(s1);

    [mag1 b]=max(abs(X1(2:end)));

    X2=fft(s2);
    [mag2 c]=max(abs(X2(2:end)));
```

```
 bode(m,1)=fcont ;
    bode1(m,1)=fcont ;
    bode1(m,3)=mag1 ;
    bode1(m,4)=mag2;

 %calculo el modulo del bode
 modg=20*log10(abs(mag2/mag1));
 bode1(m,2)=modg;

 if m==1
    modg1=modg;

 else
     %eliminamos errores de calculo tipo NaN
     if isnan(modg)
         if m>3
         modg1=mean(bode(m-3:m-1,2));
         else
             modg1=modAnt;
         end

     end

    %filtramos los datos siempre que esten dentro del rango +-R

     if modg<(modAnt+R*abs(modAnt)) && modg>(modAnt-R*abs(modAnt))
         modg1=0.5*modAnt+modg*0.5;

     else
         modg1=modAnt;
     end
 end

modAnt=modg1;


 bode(m,2)=modg1;



 m=m+1;
 j=j+per;
```

```
    f=10^j;
    fcont=round(f);
        i=i+1;

    else
       i=i+1;
    end

end



end
```

```
function [xp,yp,yz]=clasif(bode1)

%%%%%%%%%Clasificador%%%%%%%%%

%bode que queremos clasificar
xp=bode1(:,1);
yp=bode1(:,2);

%cargamos los patrones ya conocidos
load patrones

semilogx(xp,yp,'--*','LineWidth',1)

s=[];

%limite o factor de pertenencia

Pert=0.7;

npat=size(patrones);

for j=1:npat(3)
%ancho de las funciones de pertenencia
dy=2;
y=patrones(:,2,j);
mfsy=[y-dy y y+dy];
sizepro=size(y);
```

```
%evalua la gráfica experimental para ver a que patron
%tiene valor más cercano
for i=1:1:sizepro
        out(i,j)=evalmf(yp(i),mfsy(i,:),'trimf');
end

media=mean(out);

%si la media es mayor que el valor de pertenencia ese
%es un posible patron correcto, el correcto será aquel
%con mayor media
if media(j)>Pert
    s=[s 1];
sprintf('pertenece al patron %d con un valor de...
 ...pertenencia del %f \n',j+3,media(j))
else
s=[s 0];
end

yz(:,j)=y;


end

%representar los resultados
hold on
semilogx(xp,yz,'LineWidth',2)
grid
axis([200 30000 -26 6])


end
```

Bibliografía

[1] Touché, *Touché for Arduino*, `http://www.instructables.com/id/Touche-for-Arduino-Advanced-touch-sensing/` [Consulta: 31-08-2015]

[2] Sato, M., Poupyrev, I., & Harrison, C. (2012, May). *Touché: enhancing touch interaction on humans, screens, liquids, and everyday objects.* In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 483-492). ACM.

[3] Cheng, K. S., Chen, C. Y., Huang, M. W., & Chen, C. H. (2006). *A multi-frequency current source for bioimpedance application.* In 5th International IEEE EMBS Special Topic Conference on Information Technology in Biomedicine.

[4] Holder, D. S. (Ed.). (2004). *Electrical impedance tomography: methods, history and applications.* CRC Press.

[5] Duoandikoetxea, J. (2003). *Lecciones sobre las series y transformadas de Fourier.* Notas de Clase, UNAN-Managua.

[6] John H. Mathews and Russell W. Howell. *COMPLEX ANALYSIS: for Mathematics and Engineering.* 2012

[7] RUDIN, Walter; PIGA, A. Casal. *Análisis real y complejo.* Alhambra, 1979.

[8] Stockham Jr, T. G. (1966, April). *High-speed convolution and correlation.* In Proceedings of the April 26-28, 1966, Spring joint computer conference (pp. 229-233). ACM.

[9] PALOMO, Rogelio. *Analizadores de Señal.* Instrumentación Electrónica. ETSI, Sevilla. Enero de 2015

[10] GONZÁLEZ, Juan Antonio. *La NDFT y su aplicación al diseño de filtros digitales 2-D y a la decodificación de señales DTMF.* Proyecto Fin de Carrera, ETSI, Sevilla.

[11] FRIGO, Matteo; JOHNSON, Steven G. FFTW: An adaptive software architecture for the FFT. En *Acoustics, Speech and Signal Processing*, 1998. Proceedings of the 1998 IEEE International Conference on. IEEE, 1998. p. 1381-1384.

[12] TOLIMIERI, Richard; AN, Myoung; LU, Chao. Cooley-Tukey FFT Algorithms. En *Algorithms for Discrete Fourier Transform and Convolution*. Springer New York, 1989. p. 72-93.

[13] Mathworks, *Definición de Matlab*, `http://es.mathworks.com/products/matlab/` [Consulta: 8-4-2015]

[14] Arduino, *Introducción a Arduino*, `http://www.arduino.cc/es/pmwiki.php?n=` [Consulta: 10-4-2015]

[15] HERRADOR, Rafael Enríquez. *Guía de Usuario de Arduino*. Universidad de Córdoba, 2009

[16] Arduino en la wikipedia, *Descripción de Arduino*, `http://es.wikipedia.org/wiki/Arduino` [Consulta: 9-4-2015]

[17] Atmega328 en la wikipedia, *Descripción del chip Atmega328*, `http://es.wikipedia.org/wiki/Atmega328` [Consulta: 9-4-2015]

[18] Arduino, *Placas de Arduino*, `http://arduino.cc/en/Main/ArduinoBoardUno` [Consulta: 10-4-2015]

[19] Arduino, *Base de datos de Arduino*, `http://www.arduino.cc/en/Reference/HomePage` [Consulta: 10-4-2015]

[20] Arduino, *Librería EEPROM*, `https://www.arduino.cc/en/pmwiki.php?n=Reference/EEPROM` [Consulta: 10-4-2015]

[21] Arduino, *Datasheet del chip Atmega328*, `http://www.atmel.com/Images/doc8161.pdf` [Consulta: 9-4-2015]

[22] Arduino, *Software Arduino*, `http://www.arduino.cc/en/Main/Software` [Consulta: 10-4-2015]

[23] DE LA ROSA, Juan José González. *Tema 4. Medidores de Impedancia y parámetros de componentes pasivos.*Instrumentación Electrónica. Universidad de Cádiz

[24] VILLALBA, José Manuel, et al. *Resonancia de ondas electromagnéticas. Medida de la permitividad dieléctrica de líquidos polares.*

[25] CODES, Fernando Valcarce. *Un sencillo medidor vectorial de impedancias.* Revista española de física, 2010, vol. 24, no 3, p. 43-47.

[26] PALOMO, Rogelio. *Transductores y acondicionamiento de señal.* Instrumentación Electrónica. ETSI, Sevilla. Enero de 2015

[27] DAVOINE, Federico. *Análisis de circuitos de CA con impedancias complejas.* Facultad de Ingeniería, Universidad de la República

[28] JALÓN, D. E.; GARCÍA, Javier. RODRIGUEZ/José Ignacio, BRAZÁLEZ Alfonso. MANUAL DE MATLAB, vol. 5.

[29] BARBERO, Antonio J., *Variación de impedancias con la frecuencia en circuitos de corriente alterna*, [Documento en línea] `https://www.uclm.es/profesorado/ajbarbero/CursoAB2007/Impedancias%20vs%20frecuencia.pdf` [Consulta: 16-4-2015]

[30] BORDONS, Carlos, *Teoría de sistemas.* Universidad de Sevilla, Departamento de Ingeniería de Sistemas y Automática, 2001.

[31] PWM, Barr, M. (2001). *Pulse width modulation.* Embedded Systems Programming, 14(10), 103-104.

[32] PWM, *Información avanzada de timers y counters*, `http://sphinx.mythic-beasts.com/~markt/ATmega-timers.html` [Consulta: 16-5-2015]

[33] RAYA, A. M. (2002). *Introducción al análisis de datos difusos*

[34] HERNANDEZ, S.E. *Lógica difusa*

[35] ESCANO, Juan M., et al. *Fuzzy matching engine for non-textual authentication: A case study.* En Science and Information Conference (SAI), 2014. IEEE, 2014. p. 615-619.