UNIVERSIDAD CARLOS III DE MADRID

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA, INDUSTRIAL Y AUTOMÁTICA

Clasificación de latidos según estándar AAMI mediante Red Neuronal sobre plataforma Intel Edison

AUTOR: Alexis Martín Cruz

TUTOR: Luis Mengibar Pozo

Septiembre 2015

RESUMEN

En la actualidad el desarrollo de dispositivos "*wearables*" y la tecnología "*para llevar encima*" son uno pilares de las grandes empresas del sector electrónico. Al amparo del gran desarrollo de los sistemas electrónicos y de las conexiones inalámbricas, como el WiFi, se están diseñando e investigando por todo el mundo hoy en día cientos de aplicaciones para aplicar la tecnología "*wearable*" que nos permiten tener un reloj inteligente en la muñeca o enviar un email de camino al trabajo mientras viajamos en tren, hasta utilizar estos dispositivos en campos en los que aún no han desarrollado todo su potencial como, por ejemplo, el entorno biomédico.

Este Trabajo de Fin de Grado consiste en aplicar las posibilidades de los dispositivos "*wearables*" en dos aplicaciones biomédicas: un clasificador de latidos mediante una Red Neuronal y el cálculo del ritmo cardíaco mediante el algoritmo de detección de latidos Pan-Tompkins diseñados en Matlab. Para la primera aplicación se utilizarán las Redes Neuronales Artificiales, un tipo de computación que aún no ha desarrollado todo su potencial y que está en constante evolución, así como las enormes bases de datos de electrocardiogramas del sitio web *Physionet*. Para la segunda aplicación se utilizará el algoritmo de detección de complejos QRS (latidos) de Pan-Tompkins, uno de los programas más robustos y eficaces en este campo. Después ambas aplicaciones serán implementadas sobre la plataforma Intel Edison, un Pc del tamaño de una tarjeta SD, y se visualizarán los resultados por el ordenador.

La finalidad del sistema es doble, por un lado diseñar las aplicaciones comentadas, y por otro el proyecto pretende, que gracias a la implementación de esas aplicaciones en la Intel Edison, la plataforma demuestre tener la suficiente memoria y capacidad de cálculo para ser la base en futuros proyectos en los que se diseñe una aplicación biomédica "*wearable*" que permita desde la recogida de datos de un ECG, hasta su procesamiento y clasificación de latidos, en un dispositivo que podemos llevar encima sin apenas notarlo.

Tabla de contenido

Índice de Figuras

Índice de tablas

Capítulo 1

Introducción

En este primer capítulo se ofrece una visión global del Trabajo de Fin de Grado realizado y de la memoria redactada, así como las motivaciones que llevaron a su desarrollo y los objetivos perseguidos.

1.1. Motivación

En la actualidad el rumbo de las aplicaciones informáticas y de los dispositivos electrónicos está en constante cambio. Hace unos años estábamos acostumbrados a hacer todo sobre el ordenador de nuestra casa, en el escritorio de nuestra habitación o despacho, o como mucho tener un Pc portátil; pero hoy en día la realidad ha cambiado. La tecnología se ha vuelto "portátil" y está al alcance de nuestra mano en cualquier lugar, escuchar música por internet mientras vamos en el tren, o descargar los archivos del trabajo en nuestro *smartphone* es algo normal hoy en día y que nos ha facilitado mucho la vida, tanto en el ámbito profesional como en el personal, para el ocio.

El mundo científico, por supuesto, no es la excepción a esta regla. Hoy en día los investigadores de todo el mundo tienen en las nuevas tecnologías recursos que hace dos décadas ni se imaginarían: ya no hacen falta grandes máquinas con infinidad de cables conectados para recoger o analizar datos; la tecnología, aparte de portátil, evoluciona hacia lo inalámbrico. Puedes estar descargando los datos recogidos por un sensor de movimiento desde tu ordenador a cientos de metros o puedes conectarte con otros investigadores a distancia sin ningún problema, lo que permite que el desarrollo científico avance aún más rápido.

La última de estas tendencias tecnológicas, que lleva ya unos años en el desarrollo de las grandes empresas, son los dispositivos "*wearables*" que se puede traducir como "la tecnología que se puede llevar encima". Se trata de desarrollar dispositivos tan pequeños que podamos llevar encima, como por ejemplo un reloj inteligente, pero a que ala vez tenga una alta capacidad de procesamiento, depende de la aplicación, y que por supuesto nos permita conectarnos por internet con el mundo sin necesidad de cables. Las posibilidades de desarrollo en este sector son inabarcables: desde aplicaciones para uso personal como GPS, pulseras que monitorizan nuestro ritmo cardíaco cuando hacemos deporte; hasta aplicaciones biomédicas que pueden mejorar el seguimiento o el diagnóstico de los facultativos sobre sus pacientes.

En concreto en el campo de los microprocesadores, la evolución ha sido brutal. Hoy tenemos a nuestro alcance plataformas con micros, y toda una serie de entradas/salidas, sensores, conexiones inalámbricas, etc, que pueden caber en la palma de nuestra mano y con una capacidad de procesamiento igual o superior a los ordenadores que hace no tanto tiempo manejábamos en nuestros despachos sin tener otra opción de acceso a la informática. Los investigadores de hoy en día, y en concreto los ingenieros electrónicos, deben utilizar las grandes posibilidades que nos ofrecen estos dispositivos para crear aplicaciones que faciliten el trabajo a profesionales de todo el mundo en multitud de ámbitos de trabajo.

1.2. Objetivos

- Objetivos del proyecto

Los objetivos generales de este proyecto son:

- Crear una aplicación que clasifique los latidos de un electrocardiograma con la ayuda de redes neuronales artificiales.

- Crear una aplicación que calcule el ritmo cardíaco por minuto basado en el algoritmo de detección de latidos de Pan-Tompkins.

- Desarrollar ambas aplicaciones y validarlas en la herramienta Matlab.

- Implementar estas aplicaciones sobre la plataforma Intel Edison, aprovechando su memoria y capacidad de cálculo.

- Ejecutar ambas aplicaciones sobre la plataforma Intel Edison y enviar los resultados a un PC para ser mostrados.

- Objetivos del sistema diseñado

Aparte de crear el sistema diseñado y los objetivos propios del proyecto, hay otros objetivos a largo plazo:

- Diseñar un sistema (las dos aplicaciones creadas), que sea la base de un desarrollo posterior para la creación de un dispositivo "*wearable*" que funcione como un solo sistema y permita: recoger la señal de un ECG, digitalizarla, filtrarla y procesarla para separar los latidos; extraer las características necesarias para clasificar los latidos en tipos predefinidos, calcular el ritmo cardíaco, y enviar los resultados a un terminal externo vía WiFi.

- Demostrar que la plataforma Intel Edison tiene la memoria, tamaño y capacidad de cálculo suficientes para ser este dispositivo comentado en el objetivo anterior.

- Demostrar las altas posibilidades de desarrollo de las redes neuronales artificiales para aplicaciones biomédicas.

1.3. Estructura del documento

Para facilitar la lectura del presente documento se ofrece un breve resumen descriptivo:

- **Introducción al Trabajo de Fin de Grado**

  Se comentaran las motivación y objetivos del proyecto, así como los recursos utilizados y la finalidad.

  - Capítulo 1: Introducción

  - Capítulo 2: Descripción general del sistema

- **Fundamentos teóricos del proyecto**

  A continuación de la introducción están aquellos capítulos donde se explicaran los conocimientos teóricos necesarios para comprender el trabajo.

  - Capítulo 3: Redes neuronales artificiales

  - Capítulo 4: Base de datos MIT-BIH Arritmia

  - Capítulo 5: Algoritmo de Pan-Tompkins

- **Trabajo práctico realizado**

  En esta parte de la memoria se documenta y desarrolla todo el trabajo experimental realizado para llevar a cabo el proyecto: desde el procesado de señales, utilización de recursos software, hasta la implementación de las aplicaciones sobre la plataforma Intel Edison.

  - Capítulo 6: Procesamiento de señales MIT-BIH Arritmia en Matlab para Red Neuronal y Algoritmo de Pan-Tompkins

  - Capítulo 7: Implementación en plataforma Intel Edison

- **Conclusiones y presupuesto**

  Conclusiones finales una vez realizado el proyecto, y presupuesto.

  - Capítulo 8: Conclusiones

  - Capítulo 9: Presupuesto

Capítulo 2

Descripción general del sistema

En este capítulo se enumerarán y explicaran los recursos software y hardware que van a ser utilizados en el proyecto, así como la finalidad que tiene el sistema diseñado.

2.1. Recursos utilizados

Aparte de las herramientas comunes utilizadas para desarrollar un proyecto de cualquier tipo como puede ser el ordenador y los programas básicos de escritura Microsoft Word o la hoja de cálculo Excel, vamos a comentar los medios utilizados para la realización de este Trabajo de Fin de Grado.

2.1.1. Matlab

Matlab [1] (abreviatura de *Matrix Laboratory*) es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M) y que está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux. Entre sus prestaciones básicas se hallan: manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware [2].

Ha sido elegida en el proyecto para crear las dos aplicaciones diseñadas: una Red Neuronal clasificadora de latidos y una calculadora de frecuencia cardíaca basada en el algoritmo de Pan-Tompkins. A pesar de que hay programas específicos para la creación de redes neuronales se optó por Matlab debido a su versatilidad para convertir su código a otros lenguajes y sus capacidades de procesamiento de datos, características que son necesarias para el desarrollo del proyecto. En cuanto al algoritmo de Pan-Tompkins se desarrollo también en Matlab, gracias a las características anteriormente citadas y a que el algoritmo ya está diseñado para esta herramienta. Se ha utilizado la versión R2014b.

2.1.2. WFDB Toolbox para Matlab

PhysioNet [3] es un servicio web libre que ofrece acceso a grandes bases de datos de registros de signos o señales fisiológicas (*Physiobank*) y a software de código abierto relacionado (*PhysioToolkit*), puesto que el uso eficaz de datos de *PhysioNet* requiere software especializado. Este "*big data*" es ya el presente de la investigación en el campo de la medicina permitiendo colaboración entre los 45000 especialistas que visitan la web cada mes.

Physiobank contiene más de 40000 registros de anotaciones organizados en 60 bases de datos. Posee aplicaciones como *LightWave* o *PhysioBank ATM* que permiten la visualización de los datos y anotaciones de los registros de todas las bases de datos. Estas herramientas han sido utilizadas para la visualización de las señales de la base de datos MIT-BIH Arritmia y han servido de apoyo al proyecto.

PhysioToolkit es una gran biblioteca de software para el procesamiento y análisis de señales fisiológicas, detección de eventos, caracterización de las señales, creación de nuevas bases de datos, etc, que esta en constante crecimiento [4], [5]. Contiene el WFDB (*WaveForm DataBase*) *Software Package*, cuyos componentes principales son la biblioteca WFDB, alrededor de 75 aplicaciones WFDB para procesamiento de señales y el *WAVE software* para la visualización y anotación de datos. Este paquete, actualizado con frecuencia, está escrito en código C altamente portátil y se puede utilizar en plataformas como GNU, Linux, MS-Windows, Mac Os X, entre otras. Además *PhysioToolkit* contiene la *WFDB Toolbox* para Matlab en código m, que contiene la mayoría de las bases de datos y aplicaciones del *WFDB Software Package* pero ejecutables en Matlab.

En este proyecto se ha decidido utilizar la *WFDB Toolbox* para el procesamiento y análisis de las señales en Matlab. Las funciones y aplicaciones utilizadas se detallan en el capítulo 6, donde se explica el proceso de trabajo realizado en el proyecto.

2.2.3. Plataforma Intel Edison

Intel Edison [6] [7] (Figura 1) un dispositivo presentado por Intel a principios de 2014, que puede definirse como un ordenador completo del tamaño de una tarjeta SD, unos 32x24x2,1 mm. Este PC en miniatura está enfocado a un mercado concreto: *wereable technology*, o la informática “para llevar encima”. Se trata de una línea de innovación de las grandes compañías y del desarrollo de productos y programas que ya no entienden la informática como algo estático que solo se puede realizar en un ordenador, sino como dispositivos portátiles que podemos llevar encima y realicen las funciones de un PC. Las herramientas de hoy en día, como el Wifi, nos permiten estar conectados al red sin necesidad de cables y por ellos los “*wereables*” son el futuro de la informática.

Figura 1. Intel Edison Compute Module

En concreto el dispositivo que utilizaremos, Intel Edison, dispone de un SoC *Intel Atom* de 22 nm con una CPU de doble núcleo a 500 MHz y una MCU a 100 MHz. Incluye 1 GB de memoria, 4 GB de almacenamiento, WiFi de banda dual y Bluetooth 4.0. Admite 40 GPIO con múltiples opciones de configuración y permite la posibilidad de añadir sensores, por lo que las aplicaciones en las que se puede utilizar abarcan todos los campos posibles y hacen las delicias de los desarrolladores de hoy en día.

El módulo Intel Edison, en nuestro caso, ha sido montado sobre el *Intel Edison kit for Arduino* [8] (Figura 2) que es una interfaz hardware que dispone de conexiones externas para pines de entrada/salida que pueden ser configurados en una gran variedad de modos de uso: PWM, SPI, I2C, ADC. Su alimentación se realiza a través de un cable mini USB.

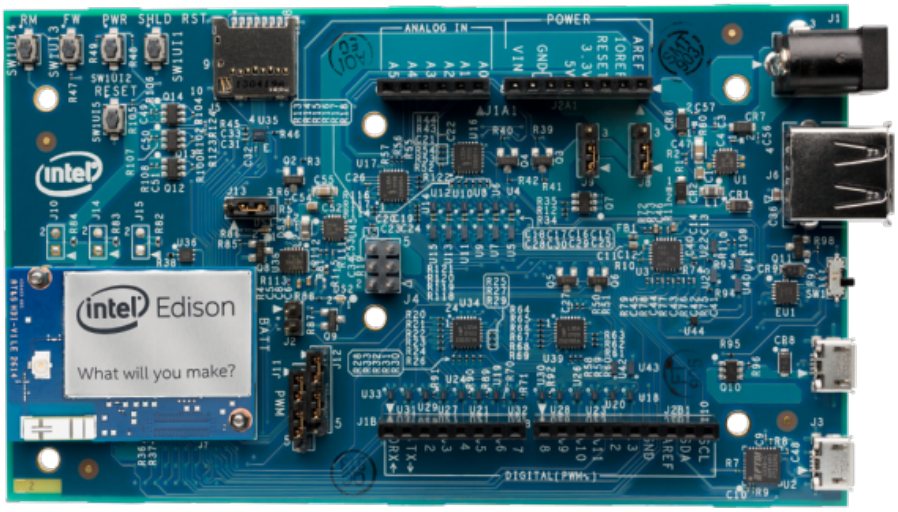


Figura 2. Intel Edison kit for Arduino

2.2.4. IDE Arduino

El entorno de desarrollo (IDE) Arduino [9] permite programar fácilmente en plataformas que soporten este código, como es el caso de nuestra Intel Edison. Arduino utiliza el lenguaje de programación del mismo nombre, Arduino, que se basa en C/C++ simplificado. Es la herramienta que se ha utilizado para programa nuestra plataforma Intel Edison. Se ha trabajado con la versión 1.5.3.

2.2. Finalidad del sistema

La finalidad del sistema diseñado es doble, por un lado se crearán dos aplicaciones comprobadas y validadas que realizan dos funciones en una plataforma de desarrollo "*wearable*": clasificar los latidos y calcular el ritmo cardíaco. Para ello nuestro sistema contará con una Red Neuronal, un sistema de computación que imita las características de funcionamiento del cerebro humano, y el algoritmo de Pan-Tompkins, un método robusto y fiable capaz de detectar los latidos dentro de un electrocardiograma (ECG). Ambas funciones serán explicadas teóricamente por separado en diferentes capítulos y después veremos como han sido diseñadas y utilizadas en el capítulo que desarrolla el trabajo realizado.

Por otra parte el sistema desarrollado en este trabajo, las dos aplicaciones anteriormente mencionadas, e implementado sobre una plataforma portátil y "*wearable*" como es la Intel Edison, está pensado para poder servir como base en futuros trabajos que amplíen lo desarrollado en este proyecto. Se pretende comprobar que la plataforma Intel Edison puede formar parte de, por ejemplo, un sistema biomédico que recoja las señales de un ECG, las procese y filtre, detecte y separe cada latido, extraiga las características necesarias, clasifique los latidos por tipos, calcule el ritmo cardíaco y envíe los resultados inalámbricamente a un terminal móvil o dispositivo electrónico. El primer paso para crear ese sistema es encontrar una plataforma que tenga la memoria y la capacidad de procesamiento necesaria para realizar todas esas funciones, por ello se quiere demostrar en este proyecto que la Intel Edison es esa plataforma que tiene la suficiente capacidad como para ser el dispositivo donde se desarrollen futuras aplicaciones como, por ejemplo, la que acabamos de describir. En definitiva, la segunda finalidad del sistema es demostrar que la Intel Edison es la plataforma adecuada para ser la innovación en los programas de análisis y diagnóstico médicos.

Capítulo 3

Redes Neuronales Artificiales

En este capítulo se explican los fundamentos teóricos de las redes de neuronas artificiales necesarios para la correcta comprensión del proyecto realizado.

3.1. Funcionamiento del cerebro humano

El cerebro humano es el sistema de cálculo más complejo que conocemos, está formado por varios billones de neuronas densamente interconectadas (Figura 3). Tareas que para un ordenador pueden ser muy complicadas, el cerebro las hace relativamente sencillas y viceversa. Esta capacidad del cerebro humano de pensar, recordar y resolver problemas muy diferentes ha inspirado a numerosos hombres de distintos campos científicos a intentar recrear las posibilidades de procesamiento y la estructura ordenada de nuestros "computadores-cerebrales" en un ordenador [10].

Figura 3. Representación de las conexiones del cerebro humano

Desde el principio del desarrollo de las computadoras se observo que permiten implementar fácilmente algoritmos para resolver problemas que antes resultaban muy largos o difíciles pero también tenían una limitación: ¿qué ocurre cuando el problema que se quiere resolver no admite un tratamiento logarítmico? Esto es debido a que las redes neuronales no aplican los principios de los circuitos lógicos o digitales, como los ordenadores que tenemos, por lo tanto debemos pensar en el cerebro como un computador analógico.

La neurona es la unidad fundamental del sistema nervioso y en particular del cerebro. Cada neurona es una simple unidad procesadora que recibe y combina señales desde y hacia otras neuronas [10]. Las neuronas forman parte del sistema nervioso humano, que en combinación con el sistema hormonal y los órganos de los sentidos así como los órganos efectores (músculos y glándulas), tienen la misión de recoger información, transmitirla y procesarla.

Lo que básicamente ocurre en una neurona biológica (Figura 4) es lo siguiente: la neurona es estimulada o excitada a través de sus entradas (dendritas) y cuando se alcanza un cierto umbral la neurona se dispara o activa, transmitiendo la señal hacia las salidas (axones) mediante un proceso electroquímico a través de las uniones llamadas sinapsis [11].

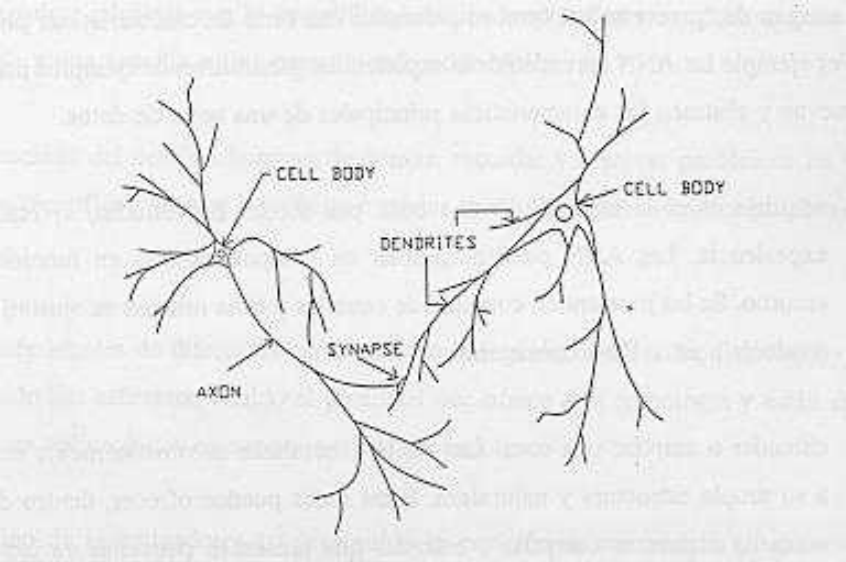


Figura 4. Partes de una neurona biológica [10]

Hay que tener en cuenta que la conectividad entre las células del cerebro es muy alta: se calcula que existen mínimo unos 100000 millones de neuronas conectadas cada una de ellas con alrededor de otras 10000 neuronas, lo que forma una red de un tamaño inimaginable. Un dispositivo de características similares es imposible de fabricar con la tecnología actual, aunque cada vez se logra una complejidad mayor en los sistemas de redes neuronales artificiales [12].

3.2. Introducción a la computación neuronal

La gran diferencia entre un programa de ordenador convencional y una máquina neuronal es que esta última elabora la información de entrada para obtener una salida o respuesta, no se trata de la aplicación ciega y automática de un algoritmo convencional. Existen muy diversos modelos, diseños, reglas de aprendizaje… para la construcción de redes neuronales artificiales pero todas parten del modelo computacional general.

La unidad análoga a la neurona biológica cerebral y fundamento de una red neuronal artificial, es el elemento procesador PE (*process element*). Un PE (Figura 5) normalmente tiene varias entradas y las combina mediante una función de suma o multiplicación ponderada, después este resultado es modificado por una función de transferencia y el valor de la salida de esta función de transferencia se pasa directamente a la salida del elemento procesador [10].

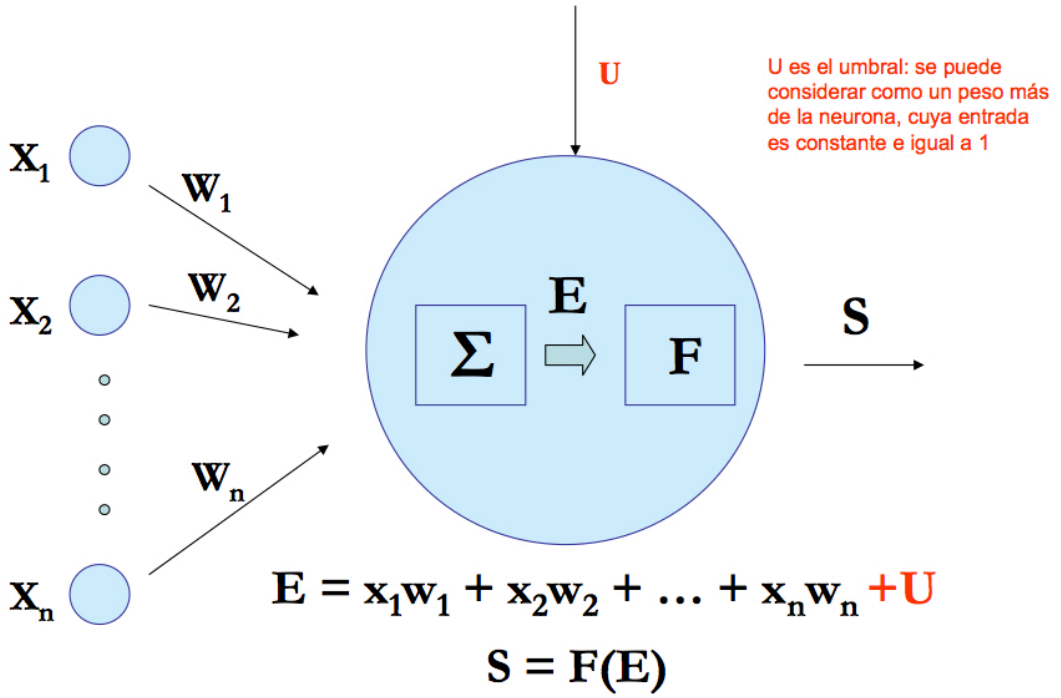


Figura 5. Partes de una neurona artificial (PE) [13]

Por lo tanto, una red neuronal artificial consiste en un conjunto de elementos procesadores PE conectados de una determinada forma. Lo interesante de las redes neuronales es que su resultado o salida no dependerá solo del modelo del elemento procesador sino de la forma en la que se conectan estos elementos. Normalmente una red típica consta de una secuencia de capas con conexiones entre capas adyacentes consecutivas, aunque hay otras arquitecturas (Figura 6). Siempre existe una capa de entrada y otra de salida, y a las capas intermedias se las llama capas ocultas [10].

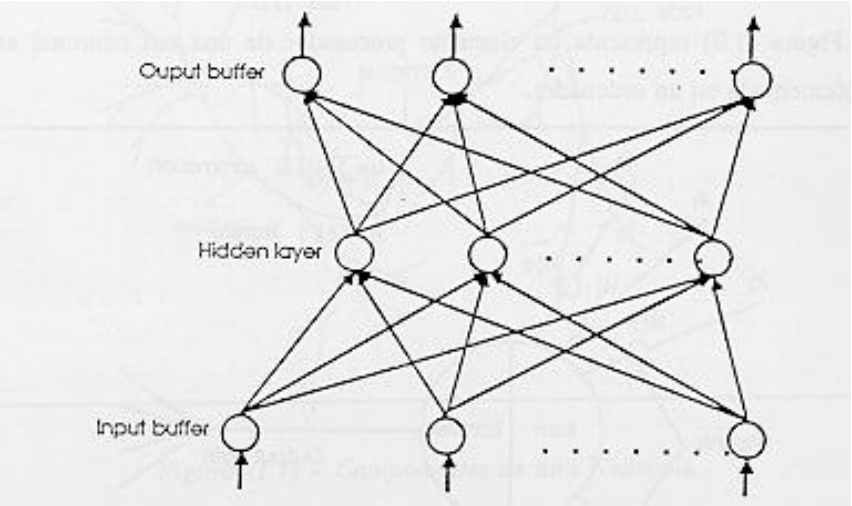


Figura 6. Capas de una red neuronal artificial [10]

La historia de la computación neuronal comienza en 1943 cuando un artículo del neurobiólogo Warren McCulloch y el estadístico Walter Pitss, "*A logical calculus of Ideas Imminent in Nervous Activity",* fue la base y el inicio del desarrollo en diferentes campos como los ordenadores digitales o la inteligencia artificial. En 1957 Frank Rosenblatt revoluciono el campo de la computación neuronal con el desarrollo del elemento "Perceptron" que sería la primera piedra para el desarrollo de diferentes arquitecturas de redes neuronales. En 1982 John Hopfield inventa el algoritmo Backpropagation que devuelve el interés a un campo que llevaba casi dos décadas de inactividad. A partir de ese momento y hasta nuestros días, los grandes avances tecnológicos han permitido que las redes neuronales artificiales sean uno de los campos donde se están realizando más trabajos de investigación y se están encontrando grandes soluciones para problemas que hasta ahora eran imposibles de resolver [11].

3.3. Fundamentos de las Redes Neuronales Artificiales

Una red neuronal artificial se puede definir como una red en paralelo de elementos simples (*process elements*) interconectada masivamente y con organización jerárquica, que intenta interactuar con las entradas-salidas de información del mismo modo que lo haría el sistema nervioso biológico [11]. Sus principales características respecto a los sistemas de computación clásicos son [10]:

- **Aprendizaje**: adquieren conocimiento por medio de la experiencia, al mostrarle un conjunto de entradas ellas mismas son capaces de ajustarse para producir salidas consistentes.

- **Generalización**: debido a su propia estructura y naturaleza, las redes que han aprendido correctamente de los patrones de entrenamiento serán capaces de responder adecuadamente antes patrones nuevos.

- **Abstracción**: algunas redes neuronales son capaces de encontrar la esencia de un conjunto de entradas que aparentemente no presentan aspectos comunes.

En los siguientes apartados se explicaran los elementos básicos de una red neuronal así como sus principales características.

3.3.1. Elementos básicos

- La neurona artificial (elemento procesador)

La neurona artificial fue diseñada para imitar las características básicas de su homologa biológica. Básicamente se aplica un conjunto de entradas a las neuronas, cada una de las cuales representa la salida de otra neurona (a excepción de la capa de entrada). Cada una de las entradas recibidas se multiplica por su peso o ponderación correspondiente, que representa el grado de conexión de la sinapsis. Todas las entradas ponderadas se suman (normalmente ya que puede haber otras funciones de entrada como el producto o el máximo) y se determina el nivel de excitación o activación de la neurona. Este nivel es procesado por una función de activación que produce finalmente la señal de salida de la neurona que será una de las señales de entrada a las neuronas de la siguiente capa (o en el caso de que nos encontremos en la capa de salida, será una de las señales de salida del sistema) [10].

Por tanto se pueden observar las similitudes entre una neurona biológica y una neurona artificial: las entradas de la neurona artificial serían las dendritas de una biológica, los pesos serían la sinapsis y las salidas serían los axones. Este tipo de neurona artificial imita pero ignora algunas de las características de las neuronas biológicas como son los retardos y el sincronismo en la salida, pero a pesar de estas limitaciones las redes neuronales artificiales presentan cualidades y atributos con cierta similitud a los sistemas biológicos (Figura 7).

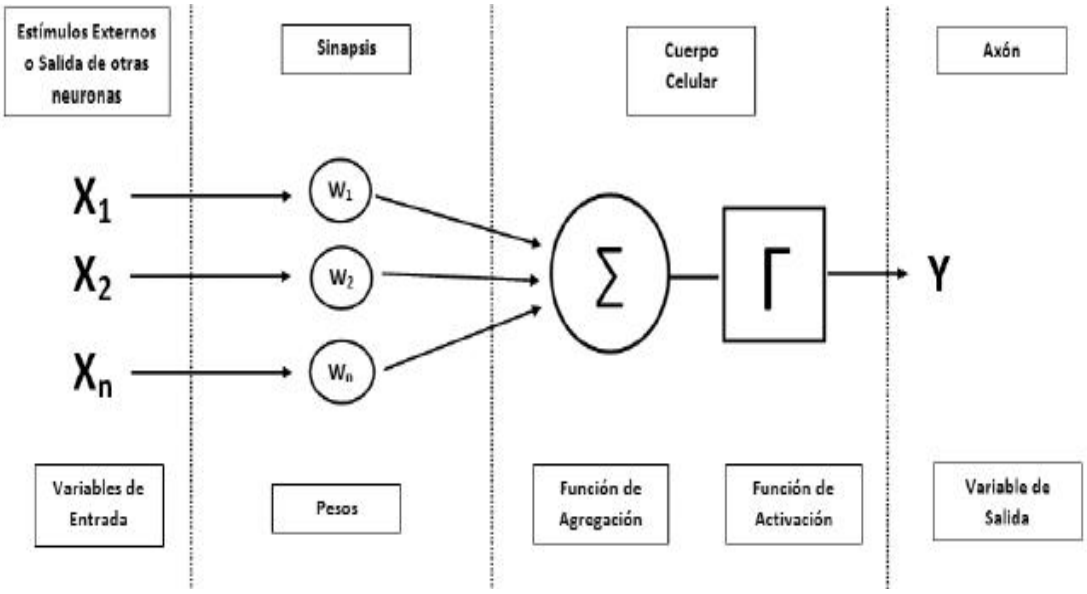


Figura 7. Analogía entre componentes de una neurona biológica y una neurona artificial

- Funciones de entrada

El elemento procesador combina muchos valores de entrada como si fueran uno solo que recibe el nombre de entrada global. Cada una de las entradas simples *(in_{i1}, in_{i2}, ...)* se combinan dentro de la entrada global, a través de la función de entrada que se calcula a partir del vector entrada:

$$input_i = (in_{i1} \bullet w_{i1}) * (in_{i2} \bullet w_{i2}) * \ldots (in_{in} \bullet w_{in})$$

donde * representa el operador apropiado (máximo, sumatoria…), *n* al número de entradas a la neurona N_i y w_i al peso.

Podemos observar que son los diferentes peso quienes cambian la medida de influencia que tienen cada uno de los valores de entrada [11].

Las funciones de entrada más utilizadas son :

- **Sumatorio de las entradas:** es la suma de todos los valores de entrada a la neurona, multiplicados por sus pesos.

$$\sum_j (n_{ij}\ w_{ij}), \quad \text{con } j = 1, 2, \ldots, n$$

- **Productoria de las entradas:** es el producto de todos los valores de entrada a la neuronas, multiplicados por sus pesos.

$$\prod_j (n_{ij}\ w_{ij}), \quad \text{con } j = 1, 2, \ldots, n$$

- **Máximo de las entradas:** solamente toma en consideración el valor de entrada más fuerte, previamente multiplicado por su peso.

$$\operatorname{Max}_j (n_{ij}\ w_{ij}) \quad \text{con } j = 1, 2, \ldots, n$$

- Función de activación

Las neuronas biológicas pueden estar activas (excitadas) o inactivas (no excitadas). Las neuronas artificiales imitan este comportamiento mediante un estado de activación que puede ser de solo dos tipos como las neuronas biológicas o también pueden tomar cualquier valor dentro de un conjunto determinado [11].

La función de activación calcula el estado de actividad de una neurona transformando la entrada global en un valor (estado) de activación. Las funciones de activación afectan al recorrido de información dentro de la red neuronal pero no a la capacidad para resolver el problema.

En la Tabla 1 podemos observar algunas de las funciones de activación más comunes:

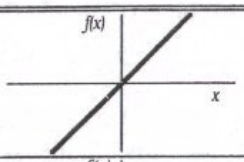
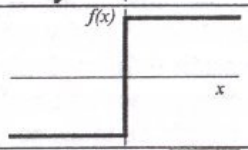
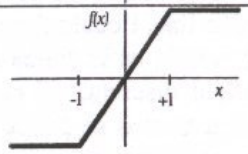
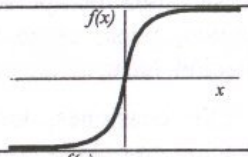
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	

Tabla 1. Funciones de activación de una neurona artificial [14]

Las funciones más utilizadas suelen ser las sigmoideas que son las que mejor comportamiento presentan. La función identidad solo suele ser utilizada en la capa de salida cuando el valor de la salida va a estar en un rango diferente al [0,1] o [-1,1]. Las funciones que tienen este rango son más comunes porque las entradas de una red neuronal suelen ser proporcionadas normalizadas en un rango para mejorar el comportamiento y aprendizaje de la red.

- Función de salida

El último componente de la neurona es la función de salida. El valor de salida de esta función será la salida de la neurona y el valor que se transfiera a la siguiente capa de neuronas, a no ser que la función de activación este por debajo de un umbral determinado, en ese caso no se pasaría ninguna salida a la neurona siguiente. Como los valores de entrada suelen estar normalizados entre [0,1] o [-1,1], los valores de salida también suelen estar comprendidos entre estos rangos [11].

Las funciones de salida más comunes son:

- **Ninguna:** la salida es igual al valor de la entrada una vez ha sido procesado por la función de activación. También se la conoce como función identidad. Es la más usada en las redes neuronales.

- **Binaria:** imita el comportamiento de la neurona biológica dando como resultado 1 si la función de activación supera el umbral, y 0 en el resto de los casos.

3.3.2. Arquitecturas o patrones de conectividad

Se denomina arquitectura de red o patrón de conectividad a la manera en que las neuronas artificiales de las distintas capas se conectan entre si (Figura 8). La distribución de las neuronas dentro de la red se realiza formando capas o niveles cada una de las cuales contiene un número determinado de neuronas. Hay tres tipos de capas [11]:

- **Capa de entrada:** Las neuronas de esta capa son las que reciben la información proveniente del exterior de la red.

- **Capa oculta:** Están formadas por las neuronas internas de la red y que no tienen contacto con el exterior. Sus entradas son las salidas de la capa de entrada, y sus salidas serán las entradas de la capa de salida. Pueden haber una capa oculta o varias (redes multicapa) o puede no haber capa oculta. Las neuronas de la capa oculta pueden estar interconectadas de distintas maneras y tener distinto número de neuronas, lo que determina su arquitectura.

- **Capa de salida:** las neuronas de esta capa transmiten la información de salida de la red hacia el exterior y son, por lo tanto, los resultados de procesar las entradas por la red neuronal.

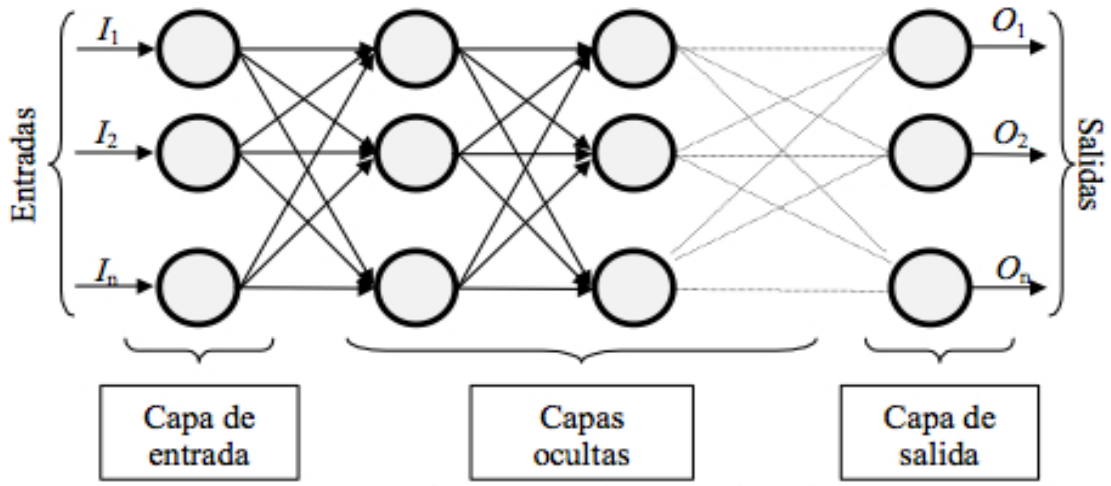


Figura 8. Esquema de una red neuronal artificial multicapa

Toda la capacidad y potencia de cálculo de una red neuronal proviene de las múltiples conexiones de neuronas artificiales que la constituyen, por ello en el diseño de una red neuronal para un problema concreto la elección del número de capas, su conexión y el número de neuronas por capa resulta fundamental para el éxito o fracaso. Normalmente una red más compleja ofrece mejores prestaciones que una red simple. Todas las posibles arquitecturas presentan conexión y aspectos diferentes pero tienen un aspecto común que imita el funcionamiento de un cerebro: el ordenamiento de las neuronas en las capas anteriormente explicadas.

Las redes multicapa se forman con grupos de capas simple en cascada conexionadas de diferente forma. Hay que tener en cuenta que si se diseña una red neuronal multicapa con función de activación lineal esta es equivalente a una red de una sola capa (Figura 9), por ello es conveniente siempre utilizar funciones de activación no lineales [10].

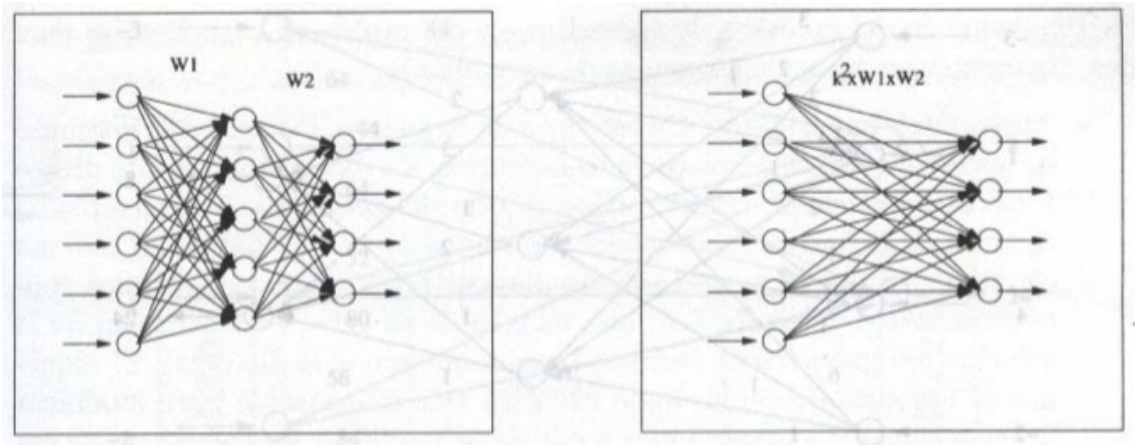


Figura 9. Equivalencia entre red multicapa con función de activación lineal y red de una sola capa [12]

Hay muchísimas arquitecturas de redes neuronales que han sido desarrolladas a partir de los primeros patrones de redes neuronales como el *Perceptron simple* o *Adaline*. Algunas de las topologías más comúnmente utilizadas hoy en día son: *Madaline*, *Perceptron Multicapa*, la red *Backpropagation*, la *Maquina de Boltzmann*, las *Memorias asociativas*, las *Redes de Hopfield*, las *Redes de neuronas de base radial*, los *Mapas auto organizados*, entre otras. Además las redes neuronales también se pueden clasificar atendiendo a su aplicación y uso. La clasificación más general que se puede hacer atiende al tipo de conexión (hacia delante, en todas direcciones, etc.) y es la siguiente [13]:

- ***1) Redes feedforward y supervisadas:*** Conexiones en un solo sentido y aprendizaje supervisado, se pueden citar *Adaline*, *Perceptron*, *Perceptron Multicapa*, *Redes de Base Radial*, entre otras.

- ***2) Redes recurrentes:*** Conexiones en todas las direcciones. Puede haber de aprendizaje supervisado y no supervisado (*Red de Hopfield*).

- **3) *Redes parcialmente recurrentes*:** Unas pocas conexiones recurrentes. Son supervisadas. *Red de Jordan*, *Red de Elman, etc.*

- ***4) Redes no supervisadas***: *Kohonen*, *ART.*

La elección del tipo de red neuronal y el diseño del número de capas y neuronas por capas no se puede hacer mediante ninguna regla, se debe realizar teniendo en cuenta el número de datos de entrada/salida, la aplicación que queremos realizar (clasificación, reconocimiento, cálculo de variables…) y la capacidad de cálculo y memoria que necesitaremos para hacer funcionar la red. En general elegir una red neuronal y sus características para una aplicación concreto es un proceso de prueba y error.

3.3.3. Mecanismos de aprendizaje

La capacidad de aprendizaje es una de las características principales y más interesantes de las redes neuronales. Es lo que las diferencias de los sistemas de computación clásicos en los que se aplica un algoritmo de resolución a ciegas. El objetivo del aprendizaje es una red neuronal es conseguir que para un conjunto de entradas la red produzca el conjunto de salidas deseadas (aprendizaje supervisado) o un conjunto de salidas mínimamente consistente (aprendizaje no supervisado) [10]. En general se puede definir el aprendizaje de una red neuronal como el conjunto de procesos en el cual los pesos de cada neurona se ajustan para dar a cada entrada la influencia correcta para lograr la salida deseada (adaptación de pesos).

El esquema de aprendizaje de una red es lo que determina el tipo de problemas que será capaz de resolver. Las redes neuronales son sistemas de aprendizaje basadas en ejemplos. También se debe tener en cuenta que la capacidad de una red para resolver un problema estará ligada de al tipo de ejemplo de los que dispone en el proceso de aprendizaje como valores de entrada. A continuación se describen los dos tipos de aprendizaje [10]:

- Aprendizaje Supervisado

En el aprendizaje supervisado se conoce de antemano los valores de salida de la red para cada uno de los patrones de entrada. Se muestran las entradas y se calcula la salida de la red, esta salida se compara con la salida deseada y el error o diferencia resultante se utiliza para realimentar la red y cambiar los pesos de acuerdo con un algoritmo que tiende a minimizar el error (Figura 10). El aprendizaje se realiza secuencialmente y de forma cíclica: se calcula el error y el ajuste de pesos para cada patrón de entrada y se repite el proceso hasta que el error sea el deseado. A continuación se explican los tres tipos de aprendizaje supervisado:

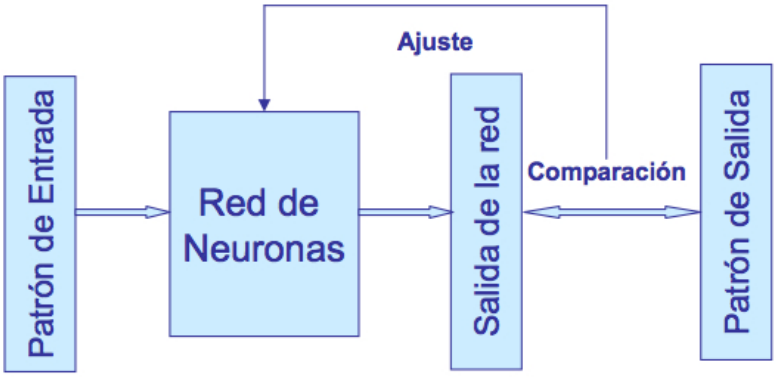


Figura 10. Esquema de aprendizaje supervisado [13]

a) Aprendizaje por corrección de error: se ajustan los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos a la salida de la red, es decir, en función del error. El ejemplo básico de este tipo de algoritmo es la regla de aprendizaje del Perceptron (1958), en donde para cada neurona de la capa de salida se le calcula la desviación a la salida objetivo como el error δ. Este error se utiliza luego para cambiar los pesos sobre la conexión de la neurona precedente (Figura 11) [11].

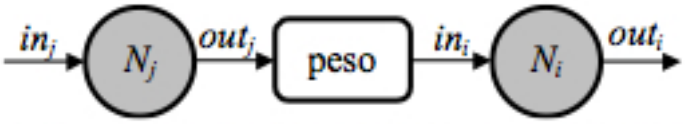


Figura 11. Influencia de la salida de la neurona N_j en la neurona siguiente N_i

Otro de los algoritmos de aprendizaje supervisado por corrección de error es la regla de aprendizaje Delta o regla del mínimo error cuadrado (LMS Error: Least Mean Squared Error), que también utiliza la desviación a la salida objetivo, pero toma en consideración a todas las neuronas predecesoras que tiene la neurona de salida, no solo a la anterior.

Por último está la regla de aprendizaje de propagación hacia atrás más conocida por su nombre en inglés *backpropagation* (regla LMS Multicapa) que es una generalización de la regla de aprendizaje Delta. Fue la primera regla de aprendizaje que permitió realizar cambios sobre los pesos en las conexiones de la capa oculta por lo que es una de los más utilizados algoritmos de aprendizaje [11].

b) Aprendizaje por refuerzo: es un tipo de aprendizaje más lento en el que durante el entrenamiento de la red no se indica exactamente la salida que se desea, simplemente se indica mediante una señal de refuerzo si la salida obtenida se ajusta a la deseada o no (éxito = 1 o fracaso = -1). En función de ello se ajustan los pesos basándose en un mecanismo de probabilidades [11].

c) Aprendizaje estocástico: consiste en realizar cambios aleatorios en los valores de los pesos de las conexiones de la red y evaluar su efecto a partir del objetivo deseado y distribuciones de probabilidad [11].

- Aprendizaje no supervisado

Los sistemas con aprendizaje supervisado han tenido éxito en multitud de aplicaciones pero desde el punto de vista biológico no son muy lógicos. Es difícil pensar que existe un mecanismo en el cerebro que compare las salidas deseadas con las salidas reales puesto que ¿de dónde provienen las salidas deseadas si el cerebro las desconoce? Por ello, los sistemas con aprendizaje no supervisado son más parecidos al aprendizaje real de un cerebro [10].

Las redes neuronales con este tipo de aprendizaje no requieren un vector de salidas deseadas y por lo tanto no realizan comparaciones entre las salidas reales y las esperadas (Figura 12). El algoritmo de aprendizaje modifica los pesos de la red de forma que se produzcan salidas consistentes, extrayendo las propiedades estadísticas del conjunto de vectores de entrenamiento y agrupando en clases los vectores similares.

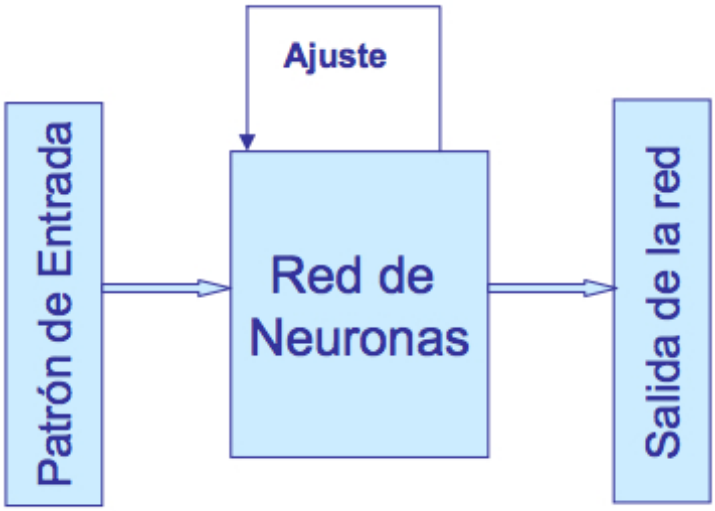


Figura 12. Esquema de aprendizaje no supervisado [13]

Hay dos tipos de aprendizaje no supervisado [11]:

a) Aprendizaje Hebbiano: esta regla es la base para muchas otras y se basa en extraer características de los datos de entrada. Su fundamento es que si dos neuronas toman el mismo estado simultáneamente, el peso de la conexión entre ambas se incrementa.

b) Aprendizaje competitivo y comparativo: se orienta a la clasificación de los datos de entrada. Su fundamento es que si un patrón de valores nuevo se determina que pertenece a una clase ya reconocida previamente, se incluye en ella. Si el nuevo patrón de valores no pertenece a ninguna de las clases anteriormente reconocidas, entonces la estructura y los pesos de la red neuronal serán ajustados para reconocer la nueva clase.

3.3.4. Entrenamiento, validación y test

Una vez se ha diseñado el tipo de arquitectura de red neuronal que queremos utilizar, el número de capas ocultas, el número de neuronas por capa y la regla de aprendizaje el siguiente paso es el entrenamiento de la red neuronal. Para realizar el entrenamiento siempre hay que seguir unos pasos previos que valen para todas las topologías y se trata de dividir y preparar los datos de los que disponemos para entrenar la red neuronal.

En primer lugar se debe determinar en la red un estado inicial, es decir, escoger un conjunto inicial de pesos para las diversas conexiones de la red. Uno de los criterios para hacerlo es otorgando un peso aleatorio a cada conexión, encontrándose todos los pesos dentro de un cierto intervalo. Durante el transcurso del entrenamiento los pesos no quedaran restringidos a dicho intervalo y a veces se tendrá que realizar una reinicialización de los pesos si el entrenamiento no se ha realizado correctamente [11].

Después de inicializar los pesos se deben codificar los datos de entrada. Si los datos de entrada no son numéricos se debe hacer una codificación y dar a cada variable un valor numérico que serán las entradas a la red neuronal. Si los datos de entrada son numéricos es conveniente realizar una codificación o normalización de los datos para transformarlos dentro del intervalo [0,1] o [-1,1].

Una vez hecho esto debemos dividir nuestros datos en tres conjuntos: conjunto de entrenamiento, conjunto de validación y conjunto de test. Los datos de entrenamiento se utilizaran para aplicar la regla de aprendizaje y modificar los pesos en función del resultado que queremos obtener. Los datos de validación se utilizan cada cierto número de ciclos de entrenamiento para analizar la evolución del error y compararlo con el error de entrenamiento, así mismo también pueden ser una condición de detención de entrenamiento como veremos más adelante. Los datos de test sirven para comprobar que después del proceso de entrenamiento la red neuronal arroja salidas correctas para nuevos datos con los que no ha sido entrenado. En general para dividir los datos se suele realizar aleatoriamente siendo el conjunto de entrenamiento un 70% del total de los datos, validación un 15% y test otro 15%. Sin embargo estos porcentajes pueden ser modificados atendiendo al tipo de datos o tipo de aplicación para el que vayamos a entrenar la red e incluso pueden no ser aleatorios y escoger datos con ciertas características para cada conjunto. En general el conjunto de datos de entrenamiento debe poseer estas características [13]:

- ***Ser significativo:*** debe haber un número suficientemente alto de ejemplo para que la red sea capaz de ajustar sus pesos de forma eficaz.

- ***Ser representativo:*** los componentes del conjunto de entrenamiento deben ser lo suficientemente diversos. Si un conjunto de entrenamiento tiene

muchos más ejemplos de un tipo que del resto, la red se especializara en dicho subconjunto y perderá capacidad de generalización (Figura 13).

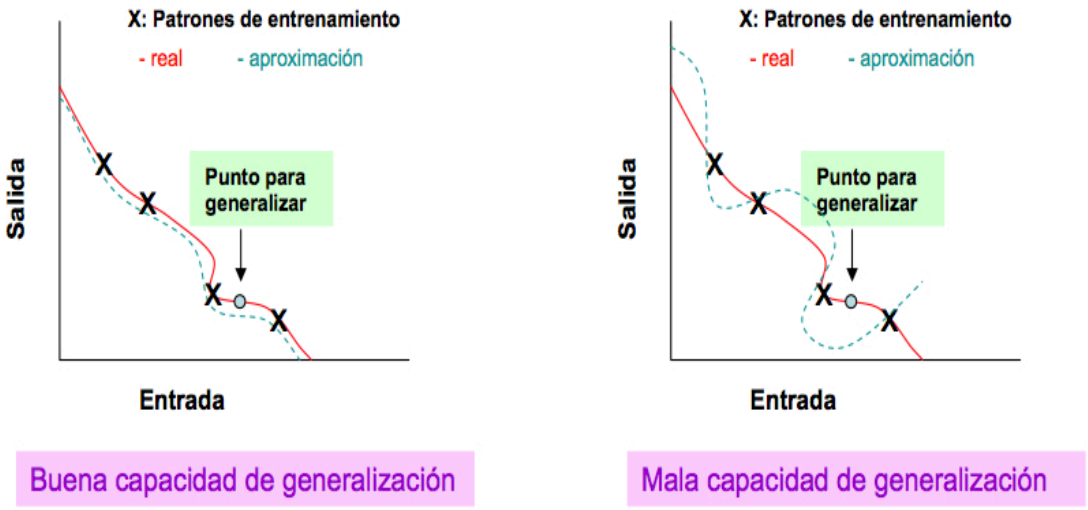


Figura 13. Capacidad de generalización en una red neuronal artificial [13]

Por último antes de comenzar el entrenamiento se debe establecer una condición de detención. Normalmente el entrenamiento se detiene cuando el cálculo del error cuadrado ha alcanzado un mínimo (aunque se puede caer en el error de detenerse en un mínimo local y no global) o cuando el error observado a la salida está por debajo de un determinado valor (Figura 14). También puede ser condición de detención cuando el entrenamiento haya realizado un cierto número de ciclos. Una vez se ha alcanzado la condición de detención, los pesos de la red no se volverán a modificar y la red está preparada para ser validada y después utilizada [11].

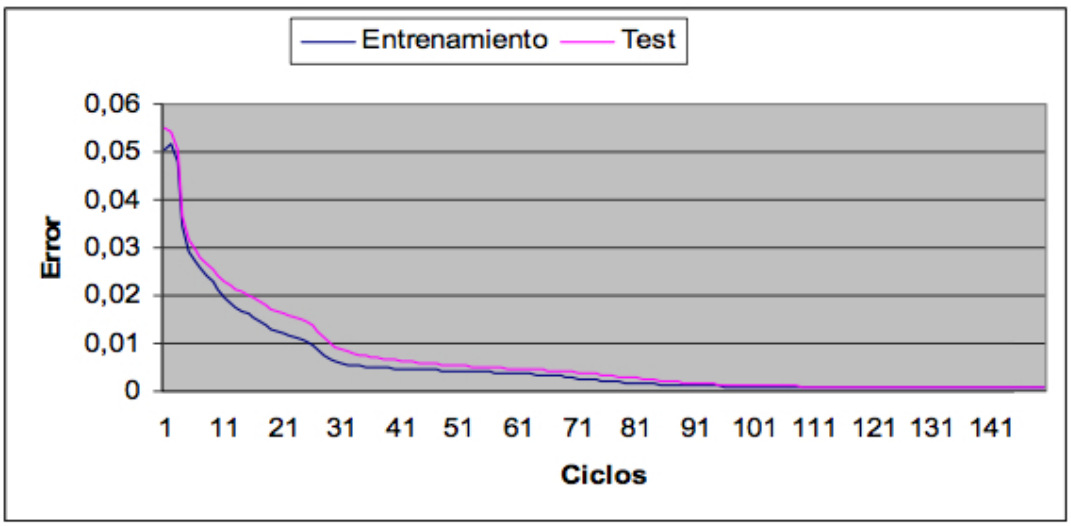


Figura 14. Grafica de evolución del error (aprendizaje con éxito) en el entrenamiento una red neuronal artificial [13]

Por último hay que tener en cuenta el problema del sobreaprendizaje (Figura 15) que se produce cuando se ha conseguido un error de entrenamiento muy bajo pero a costa de perder generalización. Esto quiere decir que la red ha sido sobreentrenada con un alto número de datos parecidos y no es capaz de realizar una respuesta adecuada para otro tipo de datos de entrada que no han sido utilizados en el entrenamiento es decir, ha perdido la capacidad de generalización y el error al utilizar los datos de test es más alto que el error de entrenamiento.

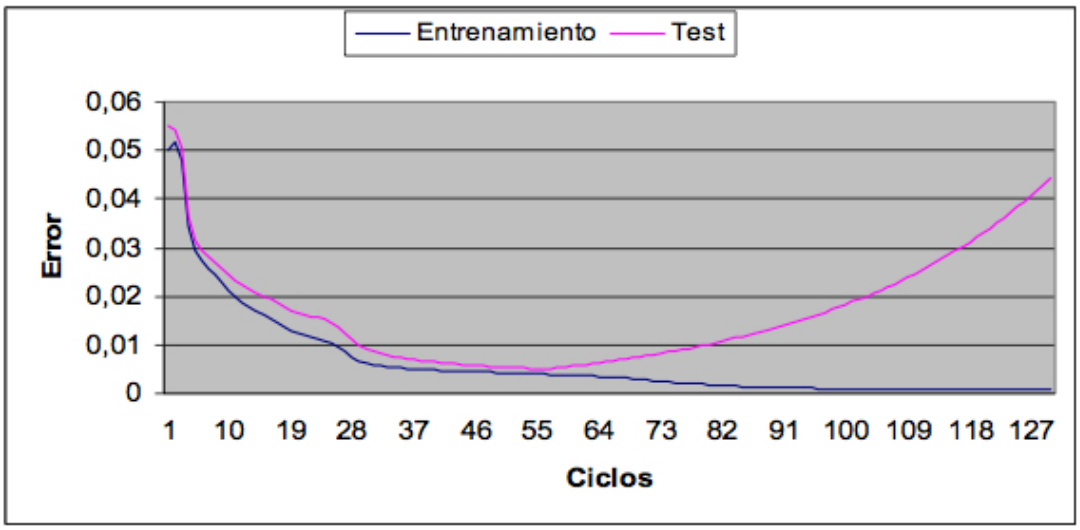


Figura 15. Grafica de evolución del error (sobreaprendizaje) en el entrenamiento una red neuronal artificial [13]

Una vez se han seguido correctamente todas las etapas de entrenamiento y el resultado de la red se ajusta a lo que hemos entrenado, el problema está en decidir cuando la salida de la red ha de considerarse como correcta. Esta es una decisión que solo puede tomar el diseñador basándose en criterios como la tasa de error o el número de iteraciones en el entrenamiento, y que dependerá de la aplicación para la que se vaya a usar la red y del nivel de exactitud que requieran los resultados.

3.4. Ventajas, inconvenientes y aplicaciones de las redes neuronales artificiales

A pesar de estar muy lejos de conseguir representar todas las características de un cerebro biológico, las redes de neuronas artificiales imitan algunas de las más útiles funciones de un sistema biológico siendo estas sus principales ventajas respecto a los sistemas de computación tradicionales [12] [13]:

- Capacidad de aprendizaje a partir de ejemplos
- Tolerancia a ruido en los ejemplos
- Diseño y modelos fáciles de utilizar
- Diversas áreas de aplicación
- Auto-organización
- Tolerancia a fallos
- Operación en tiempo real
- Fácil inserción dentro de la tecnología existente
- Representación y procesamiento distribuido de la información
- La eliminación de una neurona artificial no afecta al funcionamiento.
- Gran potencial para la construcción de sistemas que no necesitan ser programados
- No requieren de grandes procesadores para su funcionamiento

Como hemos comentado anteriormente las redes neuronales artificiales no logran imitar plenamente las características de nuestros sistemas biológicos mucho más densamente interconectados y con una estructura mucho más compleja que un simple ordenamiento por capas, por ello muestran una serie de limitaciones que enumeramos a continuación [12]:

- El tiempo de aprendizaje de la red no puede ser conocido a priori (En aplicaciones en las que se requiera procesamiento en tiempo real esto puede ser una limitación critica).
- El diseño de una red para resolver un problema con éxito puede ser una tarea muy compleja y larga, debido a que el método común para diseñar una red es mediante prueba y error.

- Comparándolas con una red neuronal biológica, las redes neuronales artificiales convergen muy lentamente y pueden necesitar de cientos o miles de patrones de entrenamiento para lograr una buena generalización.

- Las redes neuronales artificiales centran su aprendizaje y funcionamiento en la resolución de un problema en concreto, no existe una red universal para cualquier tipo de problema.

Pese a estas limitaciones las redes neuronales artificiales son, hoy en día, objeto de estudio y desarrollo por parte de muchos investigadores y universidades del mundo, puesto que gracias al alto desarrollo del hardware y las prestaciones de los computadores convencionales, las ideas y el potencial de las redes neuronales artificiales desarrolladas hace casi un siglo pueden resolver problemas que hasta ahora eran más difíciles de resolver con los sistemas de computación lógicos. Es un campo en tremendo desarrollo que ofrece muchas posibilidades en muy distintos campos de aplicación y que en los próximos años y décadas será mejorado y ofrecerá resultados que hasta ahora eran inimaginables.

Además se pueden desarrollar redes neuronales artificiales en un tiempo razonable, con la capacidad de realizar tareas concretas mejor que otras tecnologías, y que permiten la inserción de redes en sistemas ya existentes.

	Comput. convencional	RNA
Unidades de proceso	Una o muy pocas. Realiza muchas operaciones básicas	Muchas. Sólo realiza una operación básica
Poder de proceso	Núm de instrucc. por segundo	Núm de actualizaciones de pesos por segundo
Patrones	Necesita un programa detallado	No son programadas. Aprenden a partir de ejemplos
Almacenam. de información	La información se almacena en lugares específicos de memoria	Codifica la información de forma distribuida. Cada pieza de información puede estar distribuida en una parte extensa de la red Es redundante por naturaleza

Tabla 2. Comparación entre computación convencional y redes de neuronas artificiales [13]

Por todo ello como hemos dicho, los sistemas de redes neuronales artificiales tiene aplicación en prácticamente cualquier tipo de resolución de problemas, y al estar en constante desarrollo cada día aparecen más problemas donde las redes de neuronas pueden lograr una solución más rápida y mejor que los sistemas de computación clásicos. A continuación veremos los distintos campos de aplicación hoy en día de las redes neuronales [10]:

- Análisis y procesado de señales
- Reconocimiento de imágenes
- Control de procesos
- Clasificación de patrones
- Filtrado de ruido
- Robótica
- Procesado del lenguaje
- Diagnósticos Médicos
- Conversión de texto a voz
- Problemas de combinatoria
- Predicción de series temporales

Los campos de trabajo en los que pueden ser aplicadas las redes neuronales correspondes a casi todas las áreas de trabajo o investigación de los seres humanos: Biología, ámbito empresarial, medio ambiente, finanzas, medicina, militares, etc. En alguno de estos campos como la medicina o el área militar las redes neuronales nunca habían sido utilizadas debido a su poco desarrollo y a los altos grados de fiabilidad y robustez que requiere el desarrollar alguna aplicación para estos dos campos, pero gracias a la investigación de los últimos años y la complejidad en aumento de las redes neuronales se están empezando a aplicar en ellos con resultados satisfactorios. Para hacernos una idea de el estado actual de las investigaciones y del desarrollo actual en el ámbito de las redes neuronales artificiales, solo en Estados Unidos la inversión supera los 100 millones de dólares anuales [10].

Capítulo 4

Base de datos MIT-BIH Arritmia

En este capítulo se explicaran los datos, el formato y las características de la Base de datos MIT-BIH Arritmia utilizada en el proyecto, así como las características médicas de un electrocardiograma para su comprensión.

4.1. Características médicas de un latido en un ECG

La idea base de este proyecto es el análisis y clasificación de latidos provenientes de un registro electrocardiográfico, ECG o EKG; para comprender correctamente el trabajo realizado es necesario hacer una introducción sobre las características médicas que definen a cada latido y que permiten diferenciarlo y clasificarlo.

Un registro electrocardiográfico o electrocardiograma (ECG) es una prueba médica no invasiva que registra los impulsos eléctricos que estimulan el corazón y producen su contracción, así como la actividad eléctrica durante las fases de reposo y recuperación (Figura 16). El aparato de medición se denomina electrocardiógrafo. Al observar un ECG se pueden hacer 5 tipos de observaciones: frecuencia, ritmo, eje, hipertrofias e infartos [15].

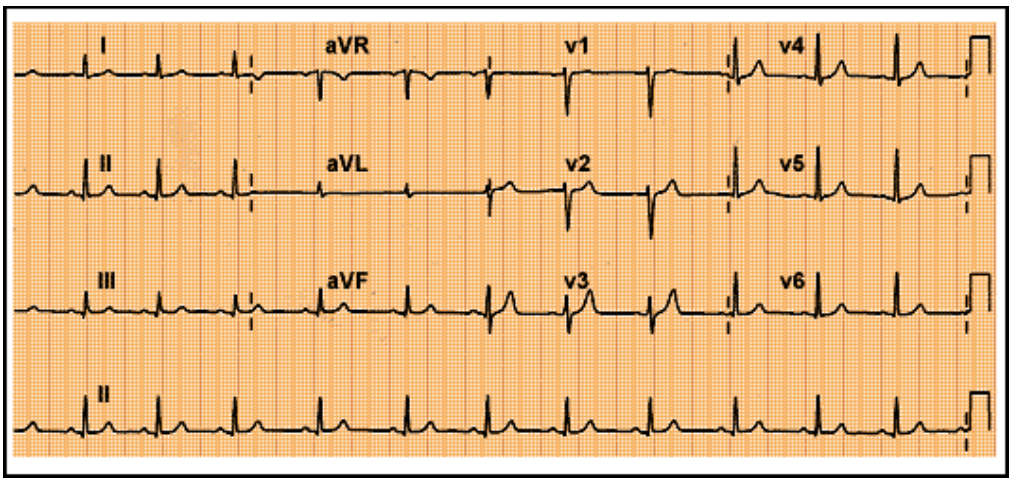


Figura 16. Ejemplo de ECG

- Derivaciones

Para la recogida de datos eléctricos por parte del electrocardiógrafo se coloca sobre la piel del paciente 10 electrodos (normalmente en los hospitales son 10 electrodos aunque se pueden hacer electrocardiogramas con menos electrodos) que consiguen 12 derivaciones básicas (con más electrodos se pueden lograr más derivaciones), es decir, 12 representaciones de la actividad eléctrica del corazón desde diferentes puntos del cuerpo. Los 10 electrodos se colocan en el cuerpo según se puede apreciar en la siguiente imagen (Figura 17).

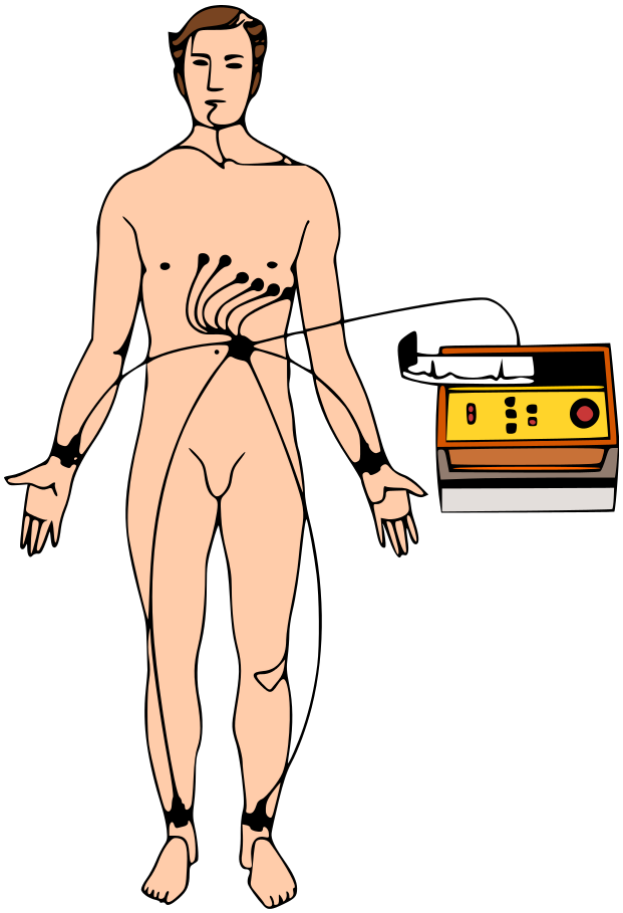


Figura 17. Colocación de 10 electrodos sobre el cuerpo, para un ECG de 12 derivaciones [2]

Los 10 electrodos colocados sobre el cuerpo del paciente se denominan de la siguiente manera:

- RA, colocado en el brazo derecho
- LA, colocado en el brazo izquierdo
- RL, colocado en la pierna derecha
- LL, colocados en la pierna izquierda

- De V1 a V6, seis electrodos colocados sobre el pecho en los espacios intercostales

De estos 10 electrodos, cada par de ellos forma una derivación [15]. Hay dos tipos de derivaciones que serán explicadas a continuación:

➢ ***Derivaciones periféricas:*** las derivaciones bipolares I, II y II miden las diferencias de potencial entre los brazos, y entre un brazo y una pierna (Figura 18); las derivaciones unipolares aVR, aVL y aVF se obtienen a partir de los mismos electrodos que las anteriores pero con la diferencia de que se mide el potencial de un electrodo del brazo con respecto a los otros dos electrodos del otro brazo y pierna unidos (Figura 19). En total estas 6 derivaciones se unen para formar 6 líneas de referencia que se cruzan en el mismo plano, es decir, es como si estuviéramos viendo la actividad eléctrica del corazón desde seis ángulos o posiciones diferentes (Figura 20). Por ello las ondas de distintas derivaciones tiene diferente aspecto, aunque representan el mismo corazón.

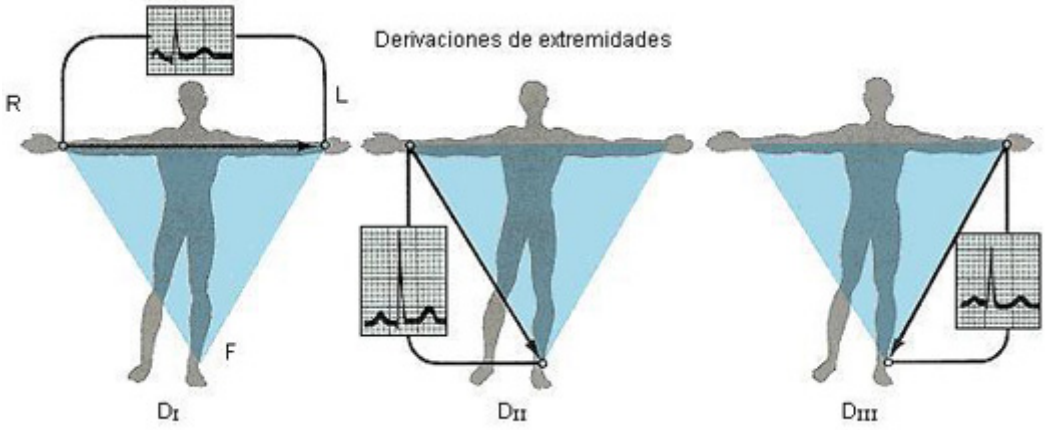


Figura 18. Derivaciones bipolares periféricas [16]

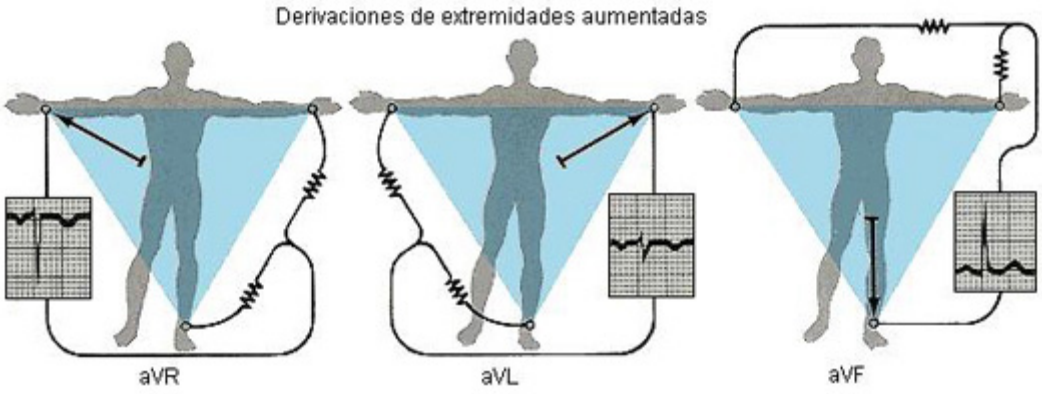


Figura 19. Derivaciones unipolares periféricas [16]

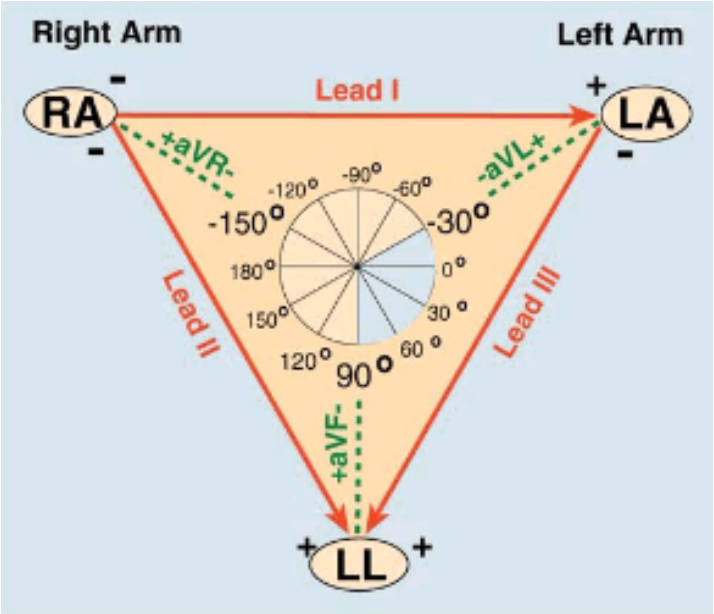


Figura 20. Derivaciones del plano central en un ECG [16]

➢ ***Derivaciones precordiales:*** las seis derivaciones unipolares precordiales se nombran como sus electrodos: V1, V2, V3, V4, V5 y V6. Los electrodos están colocados sobre el pecho y miden la diferencia de potencial respecto a un terminal central que es la media de las 3 derivaciones periféricas. Van en orden progresivo de derecha a izquierda y cubren la imagen del corazón sobre la pared torácica [15]. A diferencia de las derivaciones periféricas, las derivaciones precordiales miden la actividad eléctrica del corazón en el plano horizontal (Figura 21*).*

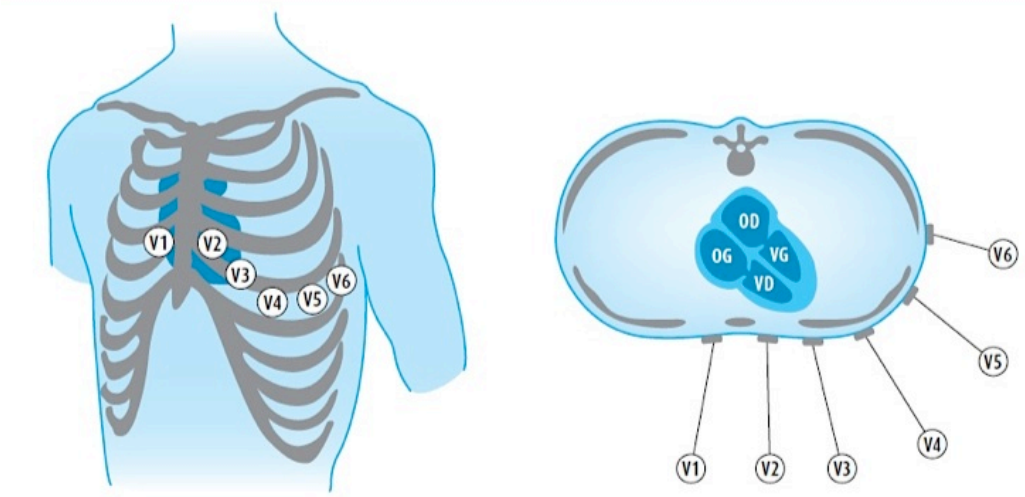


Figura 21. Derivaciones precordiales de un ECG [16]

- Medidas fundamentales del ECG

Teniendo el conocimiento suficiente y observando la forma de onda y valores de la representación gráfica de un ECG se pueden detectar y diagnosticar algunas patologías y enfermedades médicas, por ello, actualmente los desarrollos en sistemas capaces de detectar estas anormalidades en el ECG están en auge puesto que ahorrarían mucho tiempo y dinero en el sistema sanitario. Hoy en día solo hay un método fiable de revisar y diagnosticar un ECG y es que este sea revisado por un cardiólogo, por lo que tener un sistema fiable que permita diagnosticar un ECG sin la necesidad de que un humano lo revise latido a latido se ha convertido en un reto de la instrumentación médica.

Como hemos comentado en apartados anteriores en la interpretación de un ECG se pueden obtener conclusiones observando: la frecuencia, el ritmo, el eje, las hipertrofias y los infartos. Por ejemplo, observando la frecuencia, si esta es superior a unos 100 latidos por minuto, estaríamos en un episodio de taquicardia sinusal (frecuencia de impulsos cardíacos aumentada originada en el nodo S A (nodo sinoauricular, marcapasos natural del corazón). Otro ejemplo de diagnóstico en un ECG sería detectando una hipertrofia (hipertrofia es el aumento de espesor de una de las paredes de la cavidad cardíaca) en la onda P del ECG, (la onda P será explicada detenidamente más adelante) si la onda P es bifásica en la derivación V1 (tiene componente positivo y negativo) y el componente inicial de la onda es menor que el segundo componente, tenemos una hipertrofia auricular izquierda. Para ilustrar los ejemplos podemos ver en la Figura 22 un episodio de taquicardia sinusal como el que hemos descrito anteriormente, y en la Figura 23 observamos la diferencia entre una onda P bifásica donde se detecta una hipertrofia auricular izquierda y debajo una onda P normal [15].

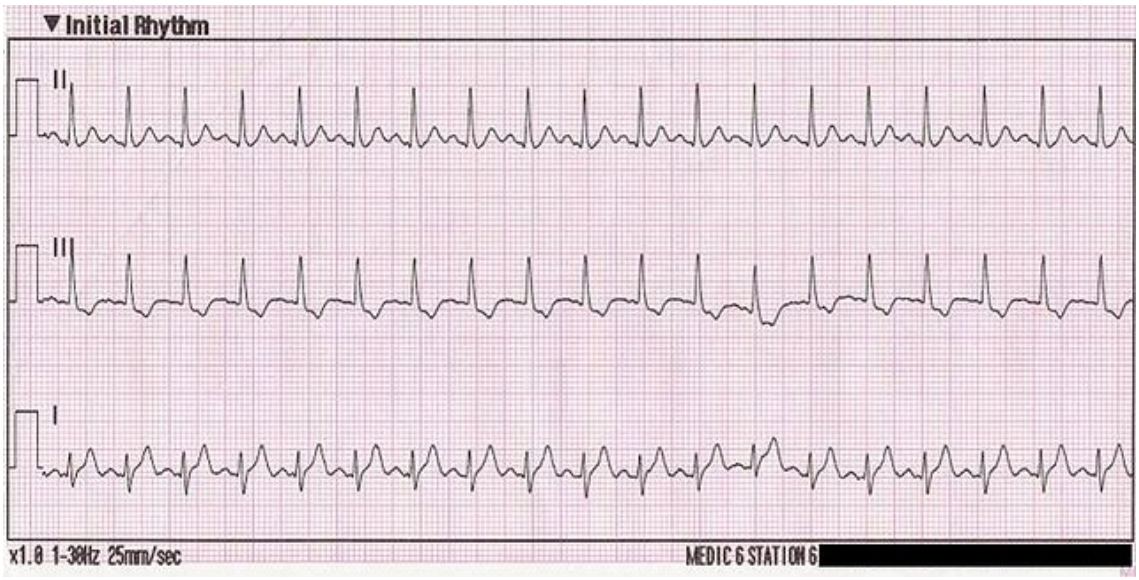


Figura 22. Episodio de taquicardia sinusal en las derivaciones I, II y III de un ECG [2]

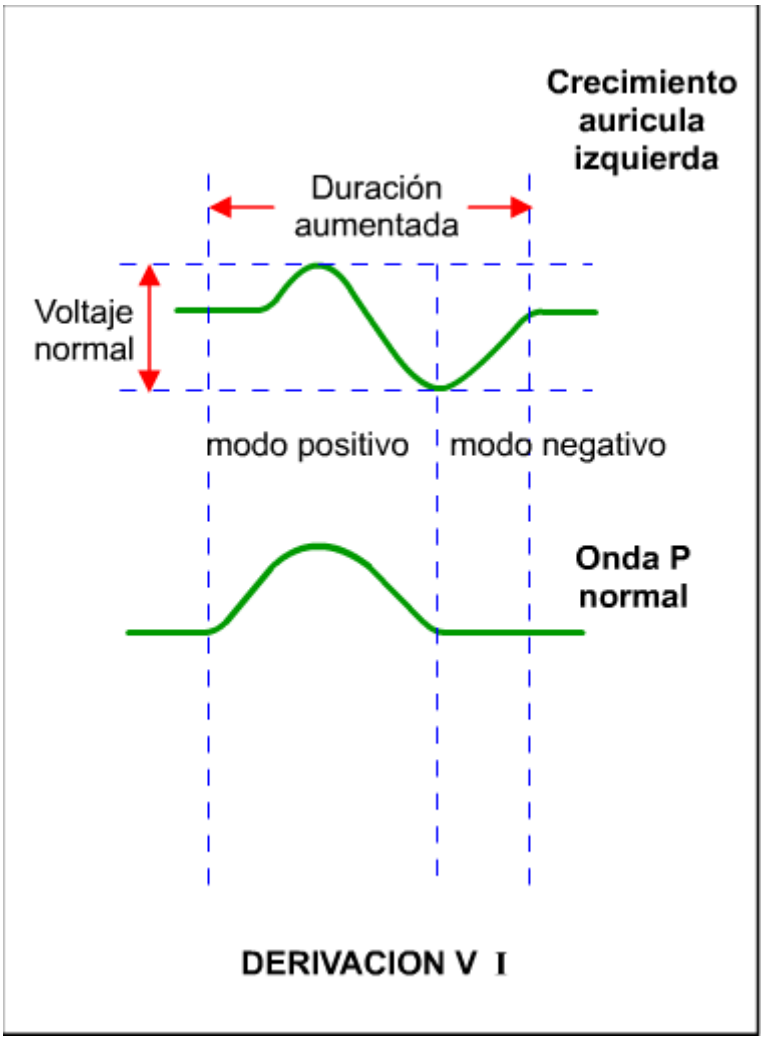


Figura 23. Onda P en hipertrofia auricular izquierda y Onda P normal (Derivación V1) [2]

Como hemos visto el estudio del ECG puede dar como resultado el diagnóstico y detección de las principales arritmias y anormalidades cardíacas, pero en nuestro proyecto nos centraremos en la detección y clasificación de latidos. Básicamente la forma de un latido humano en la representación de un ECG se puede definir como la actividad del nodo S A que produce impulsos eléctricos que se transmiten a las dos aurículas y, a través del nodo atrioventricular y el haz de His, a los dos ventrículos, siendo el resultado las contracción cardíacas. Es importante señalar que la actividad eléctrica reflejada en un latido no coincide con los fenómenos físicos que provoca esta conductividad en nuestros músculos del corazón: contracción y dilatación. A continuación vamos a explicar en detalle los componentes de un latido en un ECG, ondas, segmentos e intervalos:

- **Onda P:** Representa la despolarización auricular y precede al complejo QRS. De su observación se puede detectar algunos trastornos (hipertrofia) e incluso su ausencia puede indicar fibrilación auricular, aleteo auricular, etc.

- **Complejo QRS:** Representa la despolarización ventricular y viene después de la onda P . Su anchura en la gráfica (duración) así como su amplitud (voltaje) pueden aportar información sobre gran número de anormalidades: taquicardia ventricular, extrasístole ventricular, bloqueos de rama, etc.

- **Onda T:** Representa la repolarización ventricular y sigue al complejo QRS. Normalmente es positiva excepto en la derivación aVR y a veces en V1. Su amplitud y forma de onda (puede ser invertida o bifásica) pueden ser signos de isquemia o pericarditis entre otras patologías.

- **Onda U:** Después de la onda T, suele ser una deflexión de tan bajo voltaje que casi nunca es visible en el ECG. Corresponde a la oscilación de la repolarización de los ventrículos y su presencia está vinculada a algunos tipos de trastorno.

- **Segmento PR:** Trazo horizontal isoeléctrico (los puntos que une en la gráfica tienen el mismo potencial eléctrico) que va desde el final de la onda P hasta el principio del complejo QRS. No tiene relevancia diagnóstica.

- **Segmento ST:** Trazo horizontal isoeléctrico que va desde el final del complejo QRS hasta el principio de la onda T. Puede estar desviado del eje, si el segmento está elevado puede haber lesiones como aneurisma o pericarditis y si el segmento en descenso puede ser indicio de una lesión subendocárdica (oclusión parcial de una arteria coronaria), entre otras.

- **Intervalo PR:** Desde el inicio de onda P hasta el principio del complejo QRS. Representa la propagación del impulso inicial del nodo SA. Su excesiva duración puede denotar bloqueos, consumo de fármacos, etc.

- **Intervalo QT:** Desde el principio del complejo QRS hasta el final de la onda T. Representa la actividad eléctrica ventricular. Su duración más larga de lo normal puede indicar isquemia o síndromes hereditarios, su duración más corta indica hipercalcemia.

Los elementos explicados anteriormente son los componentes fundamentales de un latido en un ECG, aunque puede hacerse otras mediciones que aporten información adicional, pero el número mínimo de componentes para poder

conocer la naturaleza de un latido son los explicados y que se encuentran ilustrados en la siguiente imagen Figura 24. La onda U no aparece reflejada en el esquema de la figura, puesto que, a pesar de ser una de las componentes principales no suele ser visible en el ECG y por lo tanto se omite.

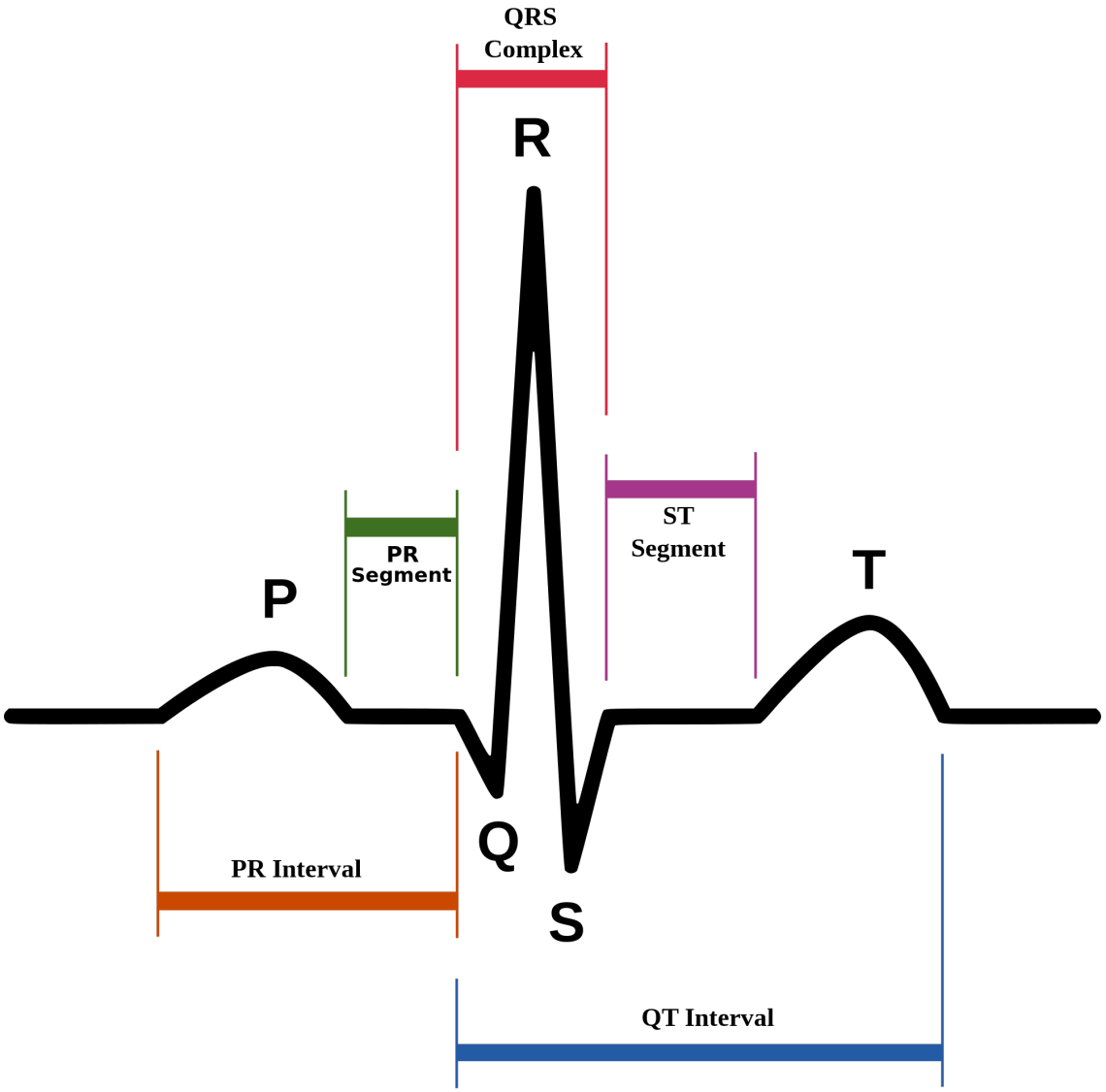


Figura 24. Componentes de un latido en un ECG [2]

En los capítulos posteriores de la memoria, donde se explicara el trabajo práctico realizado para este proyecto, se detallara que componentes de un latido se escogieron para su reconocimiento y clasificación, y porque se usaron esas características.

4.2. Señales ECG de la base de datos

MIT-BIH Arrhythmia Database [17] es una base de datos de señales de electrocardiograma elaborada por el MIT (*Massachusetts Institute of Technology, USA*) en colaboración con el BIH (*Beth Israel Hospital, now the Beth Israel Deaconess Medical Center of Boston*) desde 1975. Originalmente se comenzó a distribuir en cintas digitales de media pulgada, para acabar siendo distribuido en CD-ROM en su revisión de la tercera edición, 1992, puesta a disposición de otras entidades académicas y de investigación, por lo que ha servido como fuente de datos en más de 500 investigaciones en todo el mundo.

La base de datos contiene más de 4000 registros Holter (monitorización del registro electrocardiográfico por tiempo prolongado a una persona en movimiento) obtenidas por el BIH *Arrhythmia Laboratory* 1975 y 1979. El 60% de las grabaciones pertenece a pacientes hospitalizados y el 40% restante a pacientes no ingresados. Los primeros 23 registros (numerados del 100 al 124) están escogidos al azar de este conjunto de pacientes y los restantes 25 registros (numerados del 200 al 234) son seleccionados para incluir fenómenos clínicamente importantes pero raros que no serían bien representados en una muestra aleatoria. El primer grupo pretende servir como muestra de la variedad de formas de onda presentes en un ECG con pacientes clínicos sin enfermedades graves, y el segundo grupo lo forman los registros que contienen episodios como taquicardia ventricular o arritmias y por lo tanto puedes presentar dificultados para los detectores de arritmias. Los 48 registros que contiene la base en total son de 30 minutos y 5 segundos de duración cada uno. Los sujetos de los que se tomaron los registros fueron 25 hombres de entre 32 y 89 años, y 22 mujeres de entre 23 y 89 años.

Cada registro de ECG contiene dos señales, la primera señal es la MLII (derivación bipolar II modificada) obtenida con la colocación de los electrodos en el pecho del paciente, la segunda señal es casi siempre la V1 (derivación precordial) salvo en algunos casos en que son V2 o V5 (también derivaciones precordiales). En general la amplitud de los complejos QRS (despolarización de los ventrículos) es mayor en la MLII.

Las grabaciones analógicas originales fueron digitalizadas posteriormente con un ADC a 360 Hz (cada registro dura unos 30:05 minutos por lo que contiene 650000 muestras) con resolución de 11 bits en un rango de ± 5 mV. A pesar del proceso de digitalización y filtrado alguna de las muestras contienen ruido de baja y alta frecuencia y ruido provocado por el artefacto de reproducción analógica original (reproducidas en un *Del Mar Avionics model 660 unit*) en ambas señales (MLII y V1).

Todos los registros contienen anotaciones hechas por un detector simple de QRS, revisadas y completadas por dos cardiólogos de forma independiente, que sirven para interpretar y poder entender cada ECG. Contienen etiquetas de

cada latido (*beat*), etiquetas de ritmo cardíaco, etiquetas de calidad de la señal y comentarios. Por ejemplo, la base de datos contiene en total unas 109000 etiquetas de ritmo. Todas las etiquetas han sido revisadas y corregidas en ediciones posteriores y podemos ver un ejemplo de cómo son estas anotaciones en la Figura 25, donde observamos cinco segundos de la forma de onda de ambas señales (MLII y V1), y entre las dos señales con la etiqueta azul de referencia "atr" tenemos las anotaciones. La base de datos nos proporciona una tabla de referencia con el significado de cada una de las anotaciones para una correcta comprensión de los ECG.

Figura 25. Cinco segundos de grafica de las dos señales del registro 100 MIT-BIH Database con anotaciones [3]

Además de toda esta información, la Base de Datos MIT-BIH Arritmia nos proporciona tablas de tipos de latidos y tablas de ritmos, en las que podemos visualizar fácilmente de que tipo son los latidos de un registro o que diferentes ritmos cardíacos hay presentes en un registros. También nos proporciona información individualizada sobre cada registro así como de cada paciente mostrándonos sus características, edad y medicación (Figura 26).

Record 104 (V5, V2; female, age 66)

Medications: Digoxin, Pronestyl

Beats	Before 5:00	After 5:00	Total
Normal	65	98	163
PVC	1	1	2
Paced	197	1183	1380
Pacemaker fusion	104	562	666
Unclassifiable	5	13	18
Total	372	1857	2229

Ventricular ectopy

- 2 isolated beats

Rhythm	Rate	Episodes	Duration
Normal sinus rhythm	69-82	22	3:52
Paced rhythm	70-78	23	26:13

Signal quality	Episodes	Duration
Both clean	13	26:46
Upper noisy	5	0:24
Lower noisy	8	0:46
Both noisy	12	2:09

Notes:

The rate of paced rhythm is close to that of the underlying sinus rhythm, resulting in many pacemaker fusion beats. The PVCs are multiform. Several bursts of muscle noise occur, but the signals are generally of good quality.

Points of interest:

3:42 PVC
5:13 Noise
5:52 Transition from paced to normal sinus rhythm
6:17 Noise
8:22 Noise in lower signal
26:51 Paced, normal, and pacemaker fusion beats
29:10 Paced, normal, and pacemaker fusion beats

Figura 26. Información del registro 104 de la base de datos MIT-BIH Arritmia [3]

4.3. Tipos de latidos

Durante el último siglo, y especialmente la última mitad de siglo, los avances en el campo del diagnóstico médico han permitido a la humanidad reconocer y detectar enfermedades que anteriormente, algunas, ni siquiera se conocían los síntomas o efectos que podría tener en nuestro cuerpo. Paralelamente, el desarrollo durante el mismo periodo de los sistemas de instrumentación electrónica aplicados al campo del diagnóstico médico han posibilitado que los facultativos de todo el mundo dispongan de herramientas y aparatos electrónicos que les permiten realizar su labor con menor esfuerzo, mayor seguridad y rapidez, un ejemplo lo encontramos en libros como el de Willis J.Tompkins (desarrollador del algoritmo de Pan-Tompkins que estudiaremos en el próximo capítulo) "*Biomedical digital signal processing"* donde podemos ver todos los avances en procesamiento y tratamiento de señales aplicados a la medicina [18]. Esto repercute en un mejor y más personalizado tratamiento para el paciente, puesto que la enfermedad se diagnostica y ataja con mayor rapidez. (Ejemplo Figura 27).

Figura 27. Monitor Holter para ECG

Los avances en el campo del diagnóstico de ECG han dado como resultado una mejor y más óptima detección y clasificación de latidos. La detección de latidos se suele realizar normalmente mediante la detección de cada complejo QRS dentro del ECG. Los avances en programas que sean capaces de localizar con seguridad y exactitud los complejos QRS sin arrojar falsos negativos o falsos positivos, están a la orden del día puesto que hoy en día estos programas se puede mejorar mucho más. A continuación veremos las clasificaciones de los distintos tipos de latidos que hacen las organizaciones.

En primer lugar, la base de datos MIT-BIH Arritmia [17] clasifica los latidos en 15 tipos diferentes, dando lugar a una extensa clasificación. Para facilitar la clasificación de latidos, en este proyecto vamos a seguir la clasificación estándar de la AAMI (*Association for the Advancement of Medical Instrumentation*) según la norma ANSI/AAMI EC57:2012 [19], una clasificación universal y estandarizada que permitirá un mejor manejo de los datos. La clasificación de la AAMI contiene 5 tipos distintos de latidos. En la siguiente Tabla 3 vamos a reclasificar los tipos de latido la MIT-BIH Arritmia, en los tipos de latido de la AAMI que serán con los que trabajaremos en este proyecto.

Tipos de latido AAMI	**N**	**S**	**V**	**F**	**Q**
Descripción	**Normal Beat**	**Supraventricular Ectopic Beat**	**Ventricular Ectopic Beat**	**Fusion Beat**	**Unknown Beat**
Tipos de latido MIT-BIH	Normal Beat (N) Left Bundle Branch Block (L) Right Bundle Branch Block (R) Atrial Escape Beats (e) Nodal (junctional) Escape Beat (j)	Atrial Premature Beat (A) Aberrated Atrial Premature Beat (a) Nodal (junctional) Premature Beat (J) Supraventricular Premature Beat (S)	Premature Ventricular Beat (V) Ventricular Escape Beat (E)	Fusion of Ventricular and Normal Beat (F)	Paced Beat (P) Fusion of Paced and Normal Beat (f) Unclassified Beat (U)

Tabla 3. Reclasificación de tipos de latido de la MIT-BIH Database con los tipos de latidos ANSI/AAMI EC57:2012

Como podemos observar en la tabla los 5 tipos de latido según ANSI/AAMI EC57:2012 [19] son: Latido Normal (*Normal Beat*), Latido Ectópico Supraventricular (*Supraventricular Ectopic Beat*), Latido Ectópico Ventricular (*Ventricular Ectopic Beat*), Latido Fusionado (*Fusion Beat*) y Latido Desconocido (*Unknown Beat*). Estos 5 tipos de latido son con los que se trabajará en este proyecto para detectarlos y clasificarlos correctamente, como se explicará en los siguientes capítulos.

Capítulo 5

Algoritmo de Pan-Tompkins

En este capítulo se explicaran los fundamentos teóricos del algoritmo de Pan-Tompkins que permite la detección de latidos en un ECG, mediante la detección de los complejos QRS

5.1. Introducción

La detección con exactitud de latidos dentro de un ECG ha sido siempre uno de los grandes retos del procesamiento de señales digitales aplicadas a la medicina, puesto que el fallo en el diagnóstico de un latido puede provocar un grave perjuicio al paciente. Es por eso que, históricamente, se han desarrollado muchos sistemas de detección de latidos basados en distintos métodos, pero la ciencia médica es un campo de aplicación especial, como hemos comentado, puesto que requiere de sistemas fiables al depender de ello el tratamiento para un paciente. Hoy en día se siguen desarrollando sistemas de detección de latidos y mejorando los algoritmos ya existentes para proporcionar a nuestros facultativos una herramienta de diagnóstico que les capacite para reconocer y clasificar latidos automáticamente sin tener que revisar todo el ECG completo, aunque hoy en día aún no se ha conseguido.

La mayoría de los detectores de latidos existentes se suelen basar en la detección de los complejos QRS (uno de los componentes principales de cada latido, explicados en el capítulo anterior, capítulo 4), al ser una de las componentes del latido que posee mayor amplitud y por lo tanto, es más fácil su detección. No obstante la detección del complejo QRS puede verse dificultada ya que la señal eléctrica puede verse afectada por artefactos debido a ruido muscular, movimiento de electrodos, interferencia de línea, corrimiento de la línea de base y ondas T de alta frecuencia (Figura 28). Por ello el desarrollador de un algoritmo de detección basado en complejos QRS debe tener en cuenta todos estos factores para lograr un programa que cumpla las condiciones.

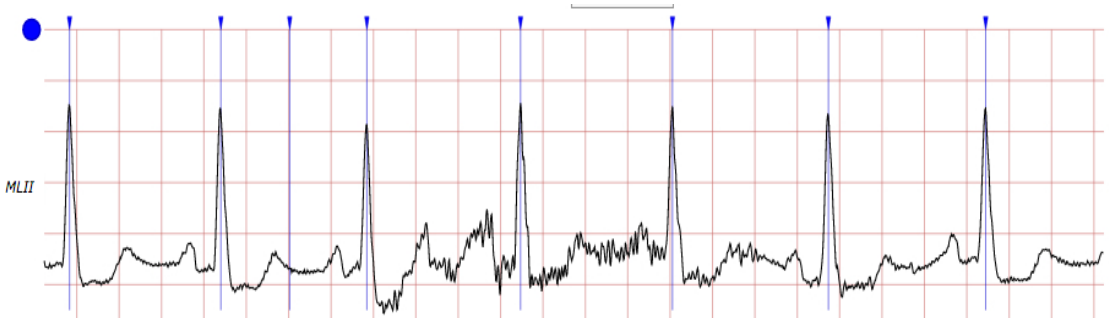


Figura 28. Señal ECG MLII con ruido [3]

Uno de los primeros sistemas de detección de latidos existentes fue el algoritmo de Holsinger de 1971 [20], que consiste en medir la tasa de cambio de la señal ECG e identificar cuando ese valor es superior a un umbral definido en un punto t, estableciendo que el punto t es parte de un complejo QRS. Sin embargo, este algoritmo tiene un alto índice de falsos positivos (cuando una parte de la señal es detectada como un latido, sin serlo realmente). Debido a las carencias de este primer algoritmo, en años posteriores se siguieron desarrollando y en 1985, Jiau Pan y Will J. Tompkins [21] presenta su algoritmo de detección de latidos. El algoritmo de Pan-Tompkins en el análisis de la pendiente, la amplitud y el ancho de los complejos QRS, para su detección

mediante una serie de etapas que se explicaran en el apartado siguiente. Al siguiente año, en 1986, Tompkins presenta, junto con Patrick S. Hamilton, los resultados de la aplicación del algoritmo mejorado de Pan-Tompkins sobre la base de datos MIT-BIH Arritmia (la misma base de datos utilizada en este proyecto, explicada en el capítulo anterior, capítulo 4), por lo que también es conocido como algoritmo de Hamilton-Tompkins [22].

5.2. Método de detección del algoritmo Pan-Tompkins

El algoritmo de detección de Pan-Tompkins es el método más robusto y confiable para la detección de complejos QRS, capaz de detectar los QRS en condiciones de mucho ruido o de señal muy alterada por causas fisiológicas. Involucra una serie de filtros y operadores de tipo paso bajo, paso alto, derivador, elevación al cuadrado, integrador, umbralización adaptiva y procedimientos de búsqueda. A continuación se detalla cada una de las etapas del algoritmo, que podemos ver en la Figura 29:

Figura 29. Etapas del algoritmo Pan-Tompkins

- ***Filtro Paso banda:*** la señal ECG donde queremos detectar los complejos QRS debe ser filtrada para obtener la banda donde se pueden encontrar los QRS y eliminar otro tipo de señales que interfieran. Primero se realiza un filtro paso bajo que elimina el ruido de alta frecuencia (ejemplo: el ruido de 50 Hz de la alimentación) y después un filtro paso-alto que elimina las componentes de continua y las ondas P y T. En la Figura 30 podemos ver la señal ECG aun sin procesar y en la Figura 31 después de haber aplicado el filtro paso banda.

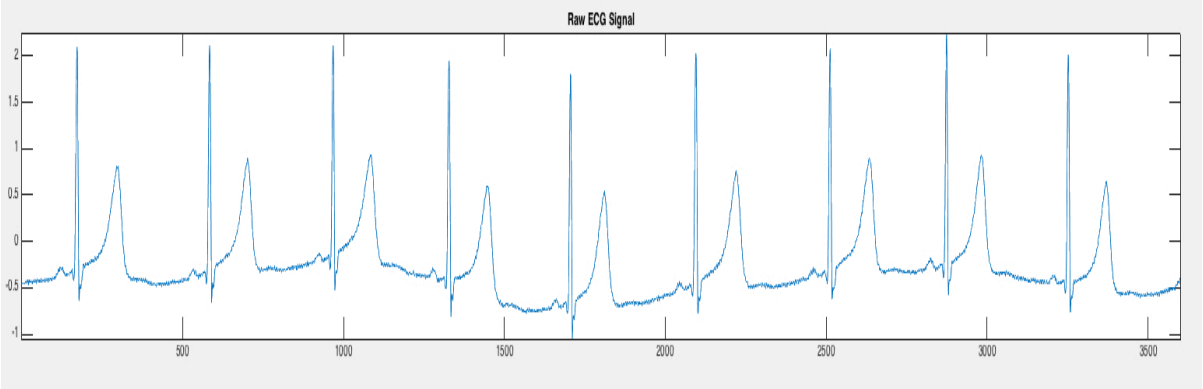


Figura 30. Señal ECG

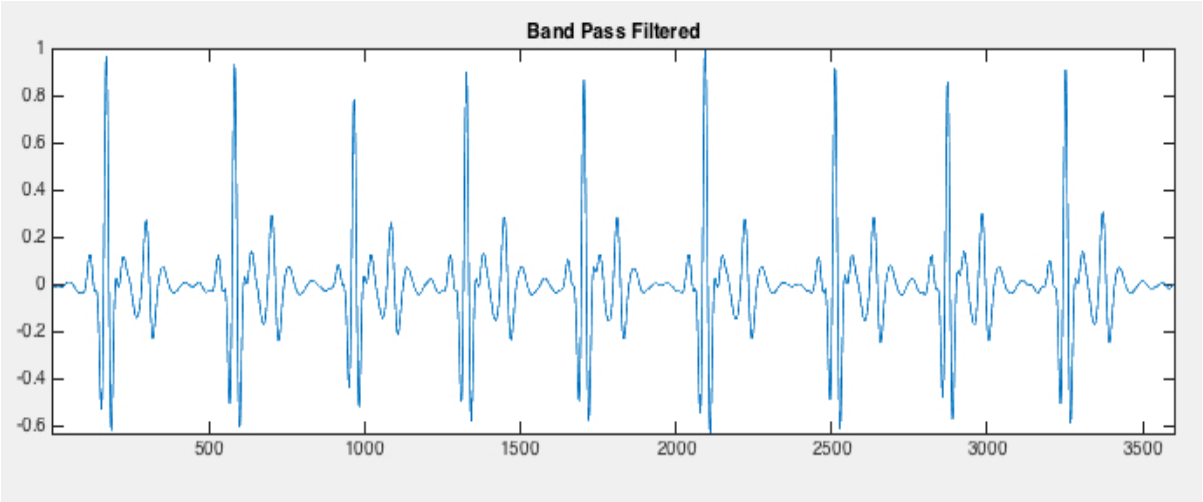


Figura 31. Señal ECG después de aplicar filtro paso banda

- ***Derivación:*** la señal es derivada para detectar las pendientes pronunciadas características de los laterales de la forma de onda del complejo QRS (Figura 32).

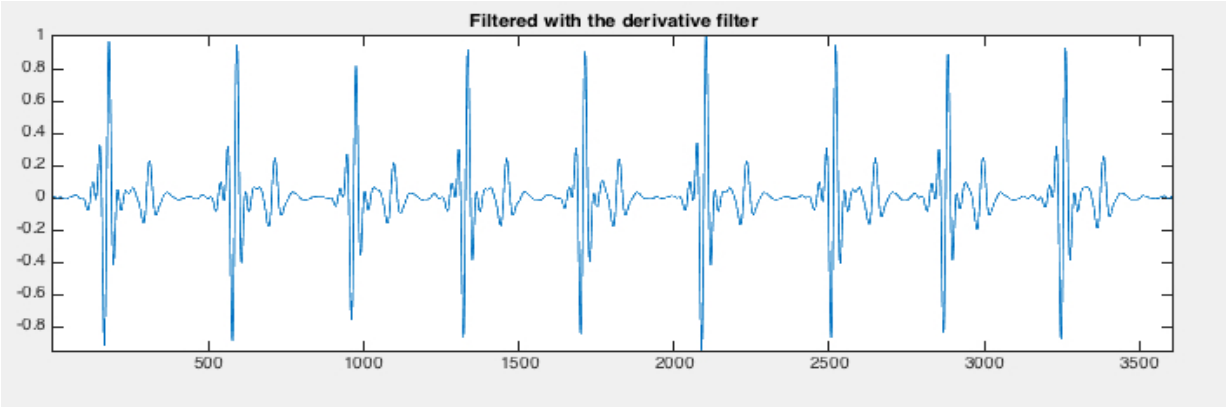


Figura 32. Señal ECG después de filtro paso banda y derivación

- ***Elevación al cuadrado:*** se eleva la señal ECG al cuadrado punto a punto, lo que genera una señal positiva intensificando las altas frecuencias y atenuando las bajas, lo que distancia los complejos QRS de las ondas T de alta frecuencia (Figura 33).

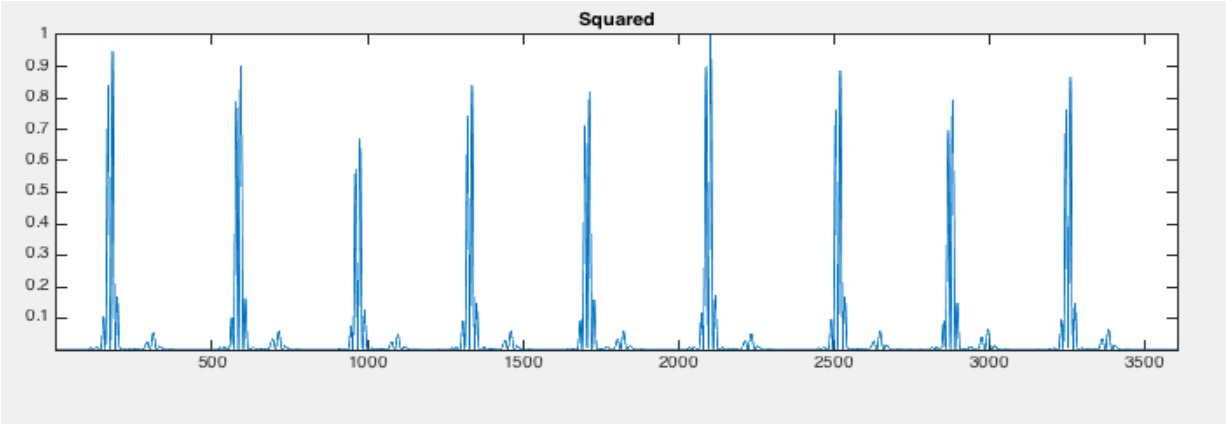


Figura 33. Señal ECG filtrada, derivada y elevada al cuadrado

- ***Integración:*** la señal debe ser integrada mediante una ventana móvil. Es muy importante que el ancho de la ventana de integración sea aproximadamente igual al ancho del complejo QRS, ya que si la ventana se queda muy pequeña o muy grande la detección no es óptima (si es más grande la ventana de integración, por ejemplo, el complejo QRS se mezcla con las ondas T). El valor de ancho de la ventana debe ser ajustado experimentalmente para cada ECG (Figura 34).

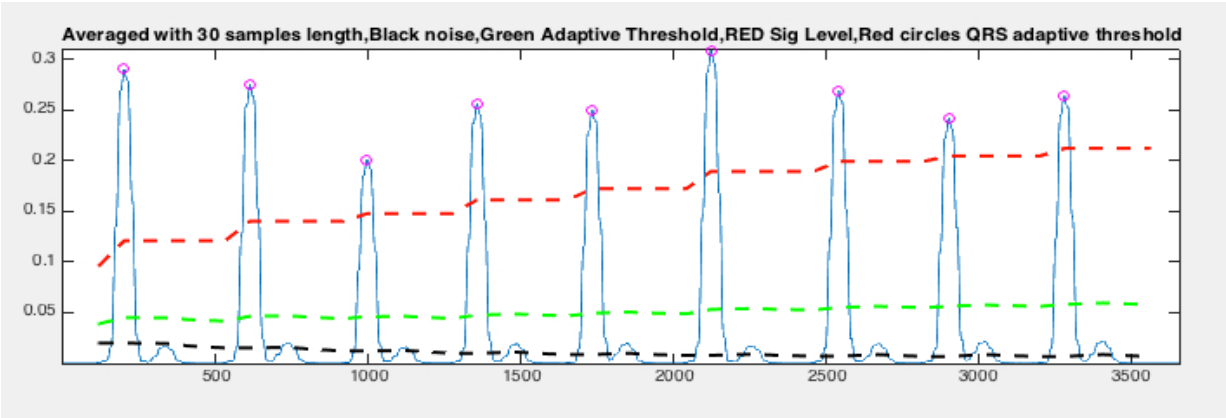


Figura 34. Señal ECG filtrada, derivada, elevada al cuadrado e integrada, con detección QRS

- ***Detección:*** si en la señal integrada se detecta un cambio de positivo a negativo, se determina un pico de energía. Después, el algoritmo, usando ***umbrales adaptativos*** determinara si este pico de energía corresponde verdaderamente a un complejo QRS o debe ser considerado ruido. Los ***umbrales adaptativos*** o estimadores calculan durante el proceso de detección los valores promedios asociados a la amplitud de los picos del complejo QRS y del nivel de ruido. En función de estos dos valores, que se van actualizando con cada nueva detección, el algoritmo decide si un pico es ruido o es un QRS. Además,

como último recurso, el algoritmo dispone de un sistema de búsqueda hacia atrás (*searchback)* que permite la búsqueda hacia atrás de un pico R cuando no ha sido detectado durante un cierto periodo de tiempo después del pico R anterior (Figura 35).

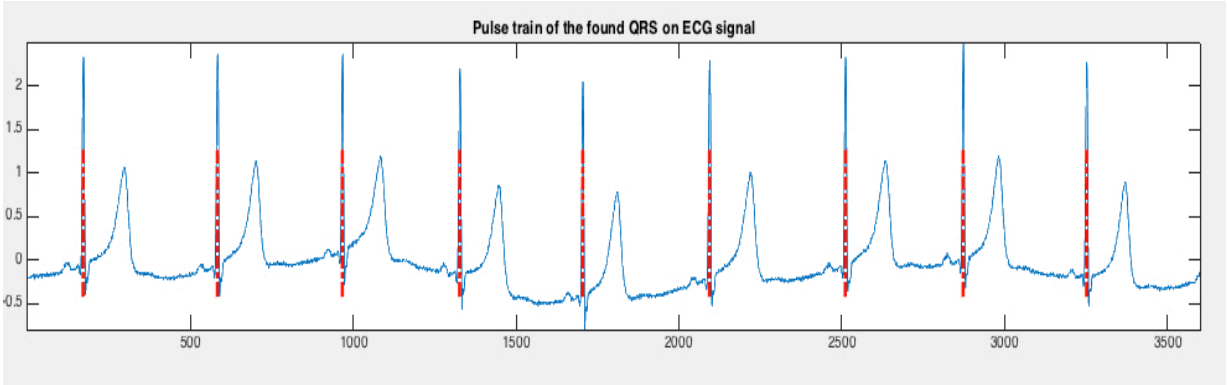


Figura 35. QRS detectados después de la umbralización y búsqueda hacia atrás

5.3. Aplicaciones prácticas del algoritmo Pan-Tompkins

Después de todo el procesamiento, el resultado del algoritmo Pan-Tompkins nos permite conocer donde están localizados los complejos QRS dentro de un ECG, es decir, donde hay un latido; por lo tanto el algoritmo es un detector de latidos. La sensibilidad del algoritmo se sitúa en un 99,69% mientras que su predicción positiva está en 99,77% [22], esto quiere decir que el algoritmo de Pan-Tompkins detecta perfectamente, o en un muy alto porcentaje, todos los latidos de una señal ECG. Un algoritmo de esta calidad que procesa y arroja resultados tan rápidos y con una fiabilidad altísima, permite que sea implementado en muchas aplicaciones que se relacionan con la detección de latidos.

En este proyecto, la finalidad del mismo es la clasificación de latidos con una red neuronal por lo que, evidentemente, deberemos usar un detector de latidos (el trabajo experimental realizado será explicado en el próximo capítulo) que podría ser el Pan-Tompkins para detectar complejos QRS. Una vez tenemos la información de donde está situado el complejo QRS de cada latido, sabemos donde están situados todos los latidos de un ECG. Esto nos permite tratar cada latido por separado y extraer características individualizadas de cada uno.

Otra de las aplicaciones médicas en las que podría implementarse el algoritmo de Pan-Tompkins sería el cálculo del ritmo cardíaco. Sabiendo donde están los latidos, y por lo tanto cuantos latidos hay en un ECG simplemente se debe saber el tiempo total de duración del ECG para realizar un cálculo del ritmo cardíaco en pulsaciones por minuto. Incluso se podría realizar un cálculo del ritmo cardíaco instantáneo conociendo la diferencia, en tiempo, de dos picos de QRS calculados por el algoritmo.

Una de las opciones más interesantes de este algoritmo es que permite su implementación en tiempo real en lenguaje C, por lo que puede ser implementado en programas de ordenador o microprocesadores. Por todo ello el algoritmo de Pan-Tompkins debido a su robustez, fiabilidad, exactitud y opciones de implementación ha alcanzado un gran reconocimiento en la comunidad científica, y son muchas las aplicaciones que se han desarrollado, y se desarrollaran en el futuro, en base a este algoritmo.

Capítulo 6

Procesamiento de señales MIT-BIH Arritmia en Matlab para Red Neuronal y Algoritmo Pan-Tompkins

En este capítulo se explicara todo el trabajo experimental realizado, desde la descarga de las bases datos, hasta el diseño de una Red Neuronal y su validación, para la clasificación de latidos; así como la implementación del algoritmo Pan-Tompkins para el cálculo del ritmo cardíaco

Primero se detallará el proceso de diseño y validación de la Red Neuronal, y posteriormente el proceso de implementación del algoritmo Pan-Tompkins

6.1. Señales de la base de datos MIT-BIH Arritmia utilizadas

El objetivo de este capítulo es desarrollar todo el trabajo realizado en la herramienta de software matemático Matlab (con la ayuda de *WFDB Toolbox* [4], [5]) con el objetivo de crear dos programas, que una vez comprobados, validados y evaluados en Matlab, serán después implementados en el microprocesador de la plataforma Intel Edison, para crear dos aplicaciones: un clasificador de latidos con Red Neuronal y una calculadora del ritmo cardíaco basada en el algoritmo de Pan-Tompkins.

El primer paso que se realizo fue elegir que bases de datos de ECG utilizaríamos. Como se explica en el capítulo 4 se escogió la base de datos MIT-BIH Arritmia [17]. El objetivo del proyecto, como se explica en el capítulo 1, es probar la implementación de una red neuronal diseñada en Matlab para la clasificación de latidos de un ECG, por lo que la base de datos que se eligiera debería cumplir las características requeridas por una red neuronal para un correcto diseño. Recordando el capítulo 3, donde se explican los fundamentos de las redes neuronales, los datos de entrenamiento de una red deben ser lo suficientemente significativos y representativos, por ello se eligió la base de datos MIT-BIH Arritmia que cuenta con un número muy alto de señales ECG y de gran diversidad, encontrando desde ritmos normales hasta señales significativas clínicamente pero raramente observadas. La base de datos escogida es una de las primeras y más utilizadas en el mundo de la investigación biomédica gracias a estas características. Para la segunda parte del proyecto, el cálculo del ritmo cardíaco utilizando el algoritmo de detección de Pan-Tompkins se utilizó la misma base de datos. A continuación se explica que registros se utilizaron para cada una de las aplicaciones, red neuronal y algoritmo de Pan-Tompkins

- ***Red Neuronal en Matlab:*** La base de datos MIT-BIH Arritmia contiene aproximadamente unos 109000 latidos repartidos en 48 registros de media hora cada uno (a 360 Hz, 650000 muestras cada registro). Cada uno de los registros descargables contiene un fichero de cabecera con información y características del registro, un archivo de datos donde está almacenada la señal, y un archivo de anotaciones donde se indica la posición de cada evento clínico dentro de la señal. Debido al tiempo de desarrollo del proyecto se decidió utilizar alrededor del 50% de latidos de la base de datos (51600 latidos), suficientes para diseñar y validar correctamente una red neuronal (si con estos datos la red neuronal no hubiera sido correctamente diseñada, se hubiera repetido el proceso aumentando el número de latidos de la base de datos). Se escogieron los registros ***100, 101, 103, 105, 106, 109, 119, 200, 202, 203, 205, 207, 208, 209, 210, 213, 214, 222, 223, 232 y 233.*** En total se han utilizado 21 registros de media hora, que han sido escogidos porque son los que contienen las muestras más significativas

y representativas de la base de datos. Es importante señalar que se ha trabajado solo con una de las dos señales procedentes de dos derivaciones del ECG (MLII Y V1, en la mayoría de registros) presentes en cada registros, en nuestro caso hemos trabajado con la señal procedente de MLII (derivaciones ECG explicadas en el capítulo 4) debido a que es la que tiene mayores amplitudes de onda. Los cuatro registros de ritmo acelerado 102, 104, 107 y 217 han sido excluidos, como en otros trabajos académicos [23], [24], [25]; y como recomienda la AAMI [26]. Recordemos que en el capítulo 4 se realizó una reclasificación de los tipos de latido de la MIT-BIH Arritmia [17] (15 tipos) a la clasificación ANSI/AAMI EC57:2012 [19] (5 tipos) como podemos ver en la Tabla 3. Ahora mostraremos en la Tabla 4 el número de latidos de cada tipo que se han utilizado en el proyecto, comparados con el número de latidos de la base de datos MIT-BIH completa:

Tipos de latido ANSI / AAMI	N	S	V	F	Q	TOTAL
Latidos utilizados en el proyecto	42.879	2.339	5.580	787	15	51.600
Base de datos MIT-BIH	90.362	2.779	7.235	803	33	109.492
% de latidos utilizados respecto del total de la base de datos MIT-BIH	47,31%	84,17%	77,15%	98%	45,45%	47,13%

Tabla 4. Número y tipo de latidos utilizados en el proyecto, comparados con el número total de latidos de la base de datos MIT-BIH

Como podemos ver en la Tabla 4 se escogieron, efectivamente, las muestras más significativas (por ejemplo, 84,17% de los Latidos Ectópicos Supraventriculares S, o el 98% de los Latidos Fusionados F). En el caso de los Latidos Desconocidos Q el porcentaje baja al 45,45% debido a que la mayoría de los latidos desconocidos Q se encuentran en una de las muestras excluidas debido a tener ritmo acelerado, la 104.

Por otro lado los datos deben ser representativos (suficientemente diversos), como se explica en el capítulo 4, y en la siguiente Tabla 5 podemos comprobar que se ha mantenido la diversidad de los datos, respetando los porcentajes de cada tipo de latido respecto al total de los

datos, es decir, que si en la base de datos total MIT-BIH los Latidos Normales N representan el 82,77% de los latidos del total (109.492) ese porcentaje se debe acercar al porcentaje de Latidos Normales N respecto al total de datos utilizados en el proyecto (51.600).

Tipo de latido ANSI /AAMI	N	S	V	F	Q	TOTAL
Latidos utilizados en el proyecto	42879	2339	5580	787	15	51600
% de tipo de latido respecto del total de latidos utilizados en el proyecto (51600)	83,1%	4,5%	10,8%	1,5%	0, 03%	100%
Base de datos MIT-BIH	90632	2779	7235	803	33	109492*
% de tipo de latido respecto del total de la base de datos MIT-BIH	82,8%	2,5%	6,6%	0,7%	0,03%	92,63%*

Tabla 5. Comparación entre los porcentajes para cada tipo de latido con respecto al total, en los datos utilizados y en la base de datos MIT-BIH Arritmia

Como vemos en la Tabla 5 los porcentajes se mantienen por lo que los datos escogidos son representativos. En el caso de los dos campos marcados con * la explicación de que los porcentajes no sumen el 100% (como sería lo normal) se debe a que no están incluidos en la suma de porcentajes los Latidos de Ritmo Acelerado P *(Paced Beat)* de las muestras excluidas (102,104,107 y 217) aunque los porcentajes si han sido calculados respecto al total de latidos de la base de datos 109492 (incluyendo los Latidos de Ritmo Acelerado P). Se ha decidido calcular así para comprobar la representatividad de los datos.

- ***Algoritmo Pan-Tompkins en Matlab:*** Para implementar el algoritmo en Matlab se decidió usar 10 s (3600 muestras) de cada uno de estos 25 registros de la base de datos MIT-BIH Arritmia: ***100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 111, 112, 113, 114, 200, 201,***

202, 203, 205, 207, 208, 209, 210, 212, 213. El algoritmo no requiere que los datos tengan ninguna características especial, simplemente ser registros digitalizados de ECGs y que se conozca su frecuencia de muestreo (360 Hz en la base de datos MIT-BIH Arritmia [17]), por lo tanto se uso la misma base de datos que utilizamos para la Red Neuronal.

6.2. Detección de latidos de cada registro ECG

Para preparar los datos de cada registro ECG de la base de datos MIT-BIH Arritmia lo primero que hay que hacer es detectar la posición de cada latido dentro del ECG, que nos permitirá saber donde están los latidos y cuantos hay. Para ello haremos uso de las aplicaciones que contiene el *WFDB Toolbox* para Matlab [4] [5], explicado en el capítulo 2, recursos utilizados. Una vez instalado el paquete en Matlab, el siguiente paso es descargar los archivos de cada uno de los registros de señal ECG (los registros utilizados están explicados en el apartado anterior).

Para encontrar los complejos QRS, es decir, los latidos; se utilizó la función **ann2rr,** que indicándole como parámetro de entrada el número de registro de la base de datos MIT-BIH, da como resultado el valor de la muestra donde está localizado el complejo QRS de cada latido (ejemplo: muestra 347) y el valor de los intervalos RR de cada latido (un intervalo RR es la diferencia entre un latido y el siguiente, intervalos entre las ondas R de dos latidos, será explicado en el apartado siguiente). Estos dos resultados de la función, contienen mucha información que se utilizara más adelante. Una vez hemos utilizado esta función en un registro tenemos la información necesaria para empezar a extraer características de cada latido.

6.3. Extracción de características de cada latido

La entrada a una Red Neuronal debe ser la información suficiente para que la red aprenda correctamente durante el entrenamiento y sea validada correctamente. En un primer momento se pensó en introducir como entrada a la red neuronal cada uno de los entre 300 y 400 valores digitales que componen un latido. Rápidamente se desecho esa idea por la cantidad de memoria que consumiría (cada registro completo son 650000 muestras) y al ser cada latido variable en muestras (un latido no dura lo mismo que el siguiente, aun en un ritmo cardíaco regular, y por lo tanto cada latido tendrá un número distinto de muestras).

Para solucionar este problema se planificó el extraer una serie de características de cada latido para que estos fuesen los datos de entrada a la red neuronal (por ejemplo: 15 datos o características de cada latido). Ahora el problema estaría en elegir cuales son las características mínimas necesarias que contienen toda la información de un latido y permiten diferenciarlo de otro para su posterior clasificación. Se decidió seguir la línea de artículos publicados por otros autores de referencia en este campo, publicados en congresos y revistas de la IEEE, ya comentados en este trabajo como [23], [24], [25], [27] y [28]. Estos artículos tratan sobre evaluación de los métodos de reconocimiento y clasificación de latidos, por lo que fueron la base para decidir las características a extraer de cada latido en nuestro proyecto. Finalmente, tras evaluar los distintos métodos de los artículos, se opto por seguir la línea de [23] y [24] y utilizar una serie de características morfológicas y valores de los intervalos de cada latido. Para elegir exactamente que valores deberían ser nuestra entrada a la red neuronal, y que esta fuese entrenada correctamente, se opto por escoger un total de 28 características presentes en ambos artículos: *intervalo pre RR, intervalo post RR, valor medio de intervalos RR, valor medio local de intervalos RR (10 valores próximos), duración del complejo QRS, duración del intervalo ST, valor booleano que indica la presencia o no de onda P y una ventana de remuestreo desde 50 ms antes del pico de onda R hasta 180 ms después del pico de onda R (en total 21 puntos de remuestreo*). A continuación pasamos a explicar detalladamente que es cada característica escogida y como se ha extraído de cada registro ECG con las aplicaciones de la WFDB Toolbox para Matlab:

- ***Intervalo pre RR:*** el intervalo pre RR es el tiempo transcurrido (en el caso de nuestra Red Neuronal los datos de entrada se refieren al número de *samples* (muestras), pero en la memoria para una mejor comprensión hablaremos de tiempo, que es simplemente el número de muestras multiplicado por el periodo, a una frecuencia de 360 Hz) entre el latido actual y el latido anterior como vemos en la Figura 36. Para extraer esta característica basta con utilizar la función de *WFDB Toolbox* para Matlab, explicada en el apartado anterior, **ann2rr** que nos da directamente como resultado el valor de todos los intervalos RR del registro.

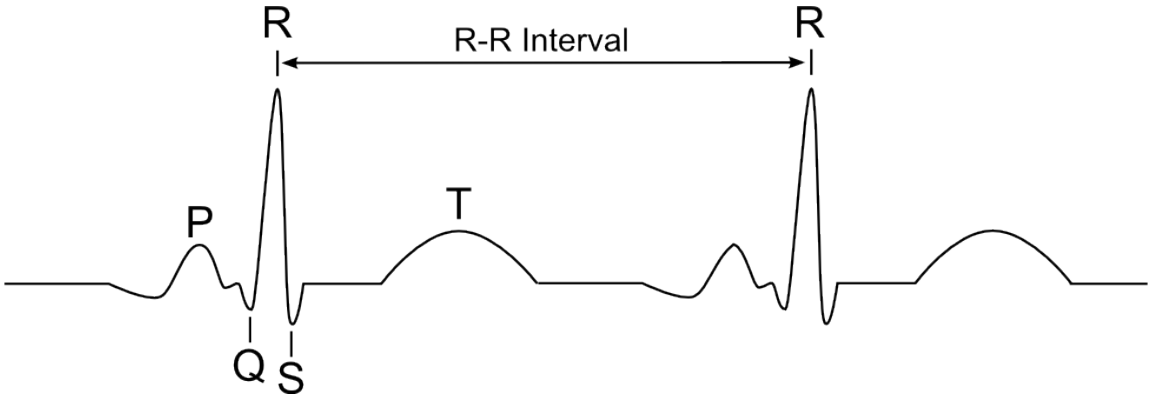


Figura 36. Intervalo RR [29]

➢ ***Intervalo post RR:*** el intervalo post es el tiempo transcurrido entre el latido actual y el latido siguiente. Es, evidentemente, un valor diferente al intervalo pre RR que mide el tiempo transcurrido hasta el latido anterior. Se extrae mediante el programa **ann2rr** (como el intervalo pre RR).

➢ ***Valor medio de intervalos RR:*** la información que aporta los valores de intervalos RR de un latido es muy limitada como para que permitan clasificarlo y distinguirlo, debido a la distinta naturaleza física de las personas cada uno tenemos nuestro ritmo cardíaco normal y será diferente al de otra persona, por ello un intervalo RR solo aporta información si se puede comparar con cual sería el valor "normal" de esa persona, es por ello que debemos conocer el valor medio de los intervalos RR de ese registro, es decir, de esa persona. Se mide haciendo la media entre todos los intervalos pre RR del registro y es el mismo valor para todos los latidos de ese registro. La media se cálculo con la función **mean2** de Matlab.

➢ ***Valor medio local de intervalos RR (10 valores próximos):*** aparte de conocer la media total de intervalos RR de ese registro, debemos conocer el valor medio de los intervalos RR próximos y así la Red Neuronal tiene los datos como para saber si el latido que está procesando tiene un intervalo RR en consonancia con su valor medio (valor medio de intervalos RR) y con el valor de los intervalos RR de los latidos cercanos (media local de intervalos RR). En datos médicos el valor medio total de intervalos RR nos da información sobre el intervalo RR "normal" de una persona y el valor medio local de intervalos RR nos da información sobre el ritmo cardíaco que tiene la persona en ese momento (ritmo normal, aleteo auricular, fibrilación auricular, etc.). Fue calculado mediante la hoja de cálculo **Excel** aplicando su función **PROMEDIO** a los intervalos pre RR de los 5 latidos anteriores y de los 5 latidos posteriores al latido en estudio.

➢ ***Duración del complejo QRS:*** la duración del complejo QRS es el tiempo transcurrido entre el inicio del complejo QRS hasta el final del complejo QRS como vemos en la Figura 37. Su duración más larga de lo normal y morfología puede indicar hipertrofias ventriculares o bloqueos de rama, entre otras patologías. Esta característica fue extraída con los siguientes programas de la *WFDB Toolbox* Matlab: en primer lugar utilizamos la función **wqrs** que crea un archivo de anotación .wqrs donde están anotados el principio y final del complejo QRS de cada latido, después con **rdann** (programa que lee un archivos de anotaciones sea .wqrs o .atr que son las anotaciones propias de cada registro) que lee el archivo de anotación .wqrs y guarda los datos en un vector de Matlab. Por último con las sentencias condicionales **if** y **for** se ha creado un programa que recorre el vector donde están los valores de inicio y fin de QRS, comprueba que la muestra donde está el pico R se encuentra entre esos dos valores (el pico R está dentro del complejo QRS) y resta el principio de onda de QRS del final de onda, dando como resultado la duración del complejo QRS y guardándolo en una nueva matriz.

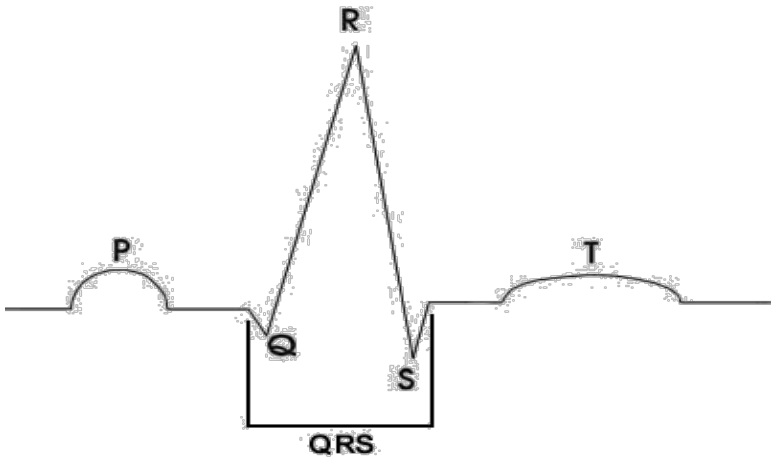


Figura 37. Duración del complejo QRS [29]

➢ ***Duración del intervalo ST:*** el intervalo ST es la suma de la duración de dos componentes fundamentales de un latido, la duración del segmento ST más la duración de la onda T, es decir (ambas explicadas en el capítulo 4) como vemos en la Figura 38. La morfología y duración del segmento ST y la onda T son fundamentales para la clasificación de un latido respecto a otro, aparte de ser necesarios para detectar enfermedades, por ejemplo el segmento ST es básico en el diagnóstico de la cardiopatía isquémica, con estudios como [30] donde se utiliza junto con una Red Neuronal para detectar esta enfermedad. Para medir esta duración se utiliza la función de *WFDB Toolbox* **ecgpuwave** que es un detector el comienzo, pico y final de las formas de onda P, QRS y T (está basado en el algoritmo de Pan-Tompkins, explicado en el apartado 5). A esta función se le puede proporcionar como parámetro de entrada las anotaciones QRS procedentes de **wqrs**

o **qrs** (un programa similar a **wqrs**, explicado anteriormente en este apartado). A continuación utilizamos **rdann** para leer las anotaciones generadas por **ecgpuwave y** guardarlas. Por último, con un programa creado con las sentencias condicionales **if** y **for** que recorre el vector donde están guardadas las anotaciones de **ecgpuwave**, comprueba que después de cada pico de onda R (latido) hay un final de complejo QRS y después un final de onda T, y guarda cada valor en distintas matrices llamadas FinQRS y FinT. Finalmente se restan ambas matrices dando como resultado una nueva matriz con los valores del intervalo ST para cada latido.

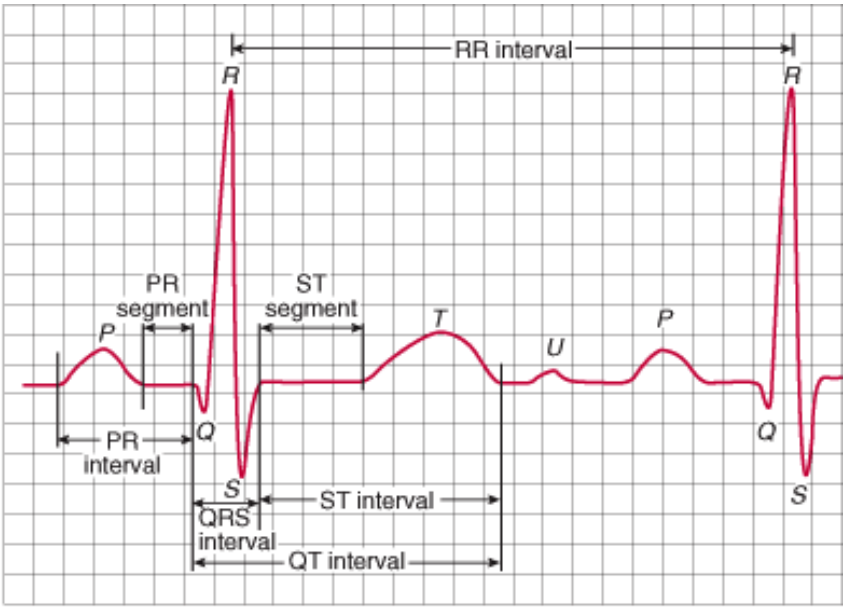


Figura 38. Componentes de un latido incluyendo el intervalo ST [31]

➢ ***Valor booleano indicando la presencia de onda P:*** es un valor 0 o 1 que nos indica si no hay onda P presente en ese latido (0) o si está presente (1). La duración, amplitud e incluso morfología (solo vale para detectar hipertrofias) de la onda P no son relevantes para el estudio de enfermedades o clasificación de latidos, pero su presencia o no en un latido es un dato importante para distinguir un tipo de latido de otro. Para comprobar si hay onda P o no en cada latido se utiliza nuevamente la función **ecgpuwave** con parámetro de entrada de anotación .wqrs (de la función **wqrs),** luego utilizamos la función **rdann** para leer las anotaciones generadas por **ecgpuwave**, pero en este caso a diferencia del anterior en vez de leer todas las anotaciones, leemos solo los picos de onda P (Figura 39) y los guardamos en un vector. Esto es posible gracias a que a la función **rdann** se le puede pasar como parámetro de entrada que tipo de forma de onda queremos leer de **ecgpuwave**: solo los picos P, o solo los inicios de onda, etc. Por último, y al igual que en los casos anteriores, con las sentencias de control **if** y **for** recorremos la matriz de picos de onda P y evaluamos si para cada valor de muestra de

latido (pico de onda R) hay antes un valor de pico de onda P, si lo hay es que ese latido tiene onda P. Finalmente generamos una nueva matriz con valores booleanos para cada latido para saber si hay o no onda P.

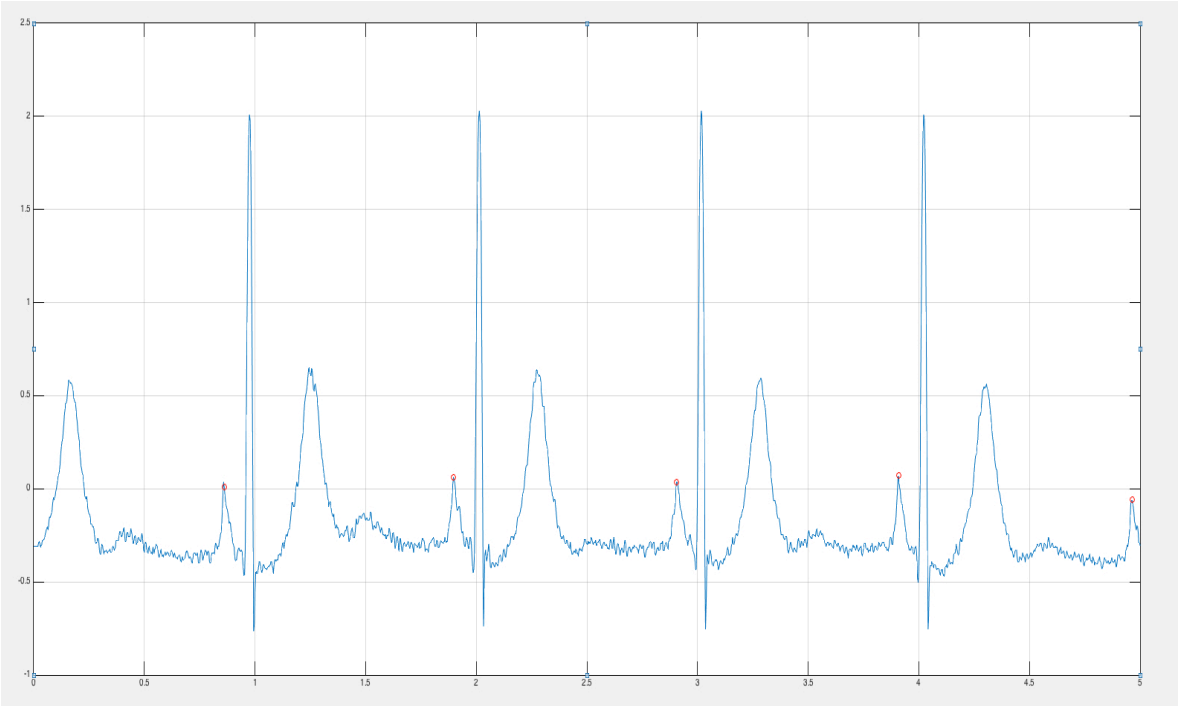


Figura 39. Detección de picos de onda P (puntos rojos) en Matlab

➢ ***Morfología del latido:*** para representar la morfología del latido se ha aplicado una ventana de remuestreo fija sobre la parte del latido que tiene importancia para la clasificación, esto es, desde el principio del complejo QRS hasta el final de la onda T (Figura 38). La ventana remuestrea desde 50 ms antes del pico de onda R, hasta 180 ms después. Estos valores han sido comprobados experimentalmente. El remuestreo se ha hecho con ayuda de la función **resample** de Matlab, indicándole un remuestreo de 10 veces menor que la frecuencia original (360 Hz) aplicando un filtro de antialiasing. El remuestreo cuenta en total con 21 muestras. En la Figura 40 podemos ver el latido de una señal original y en la Figura 41 el mismo latido remuestrado.

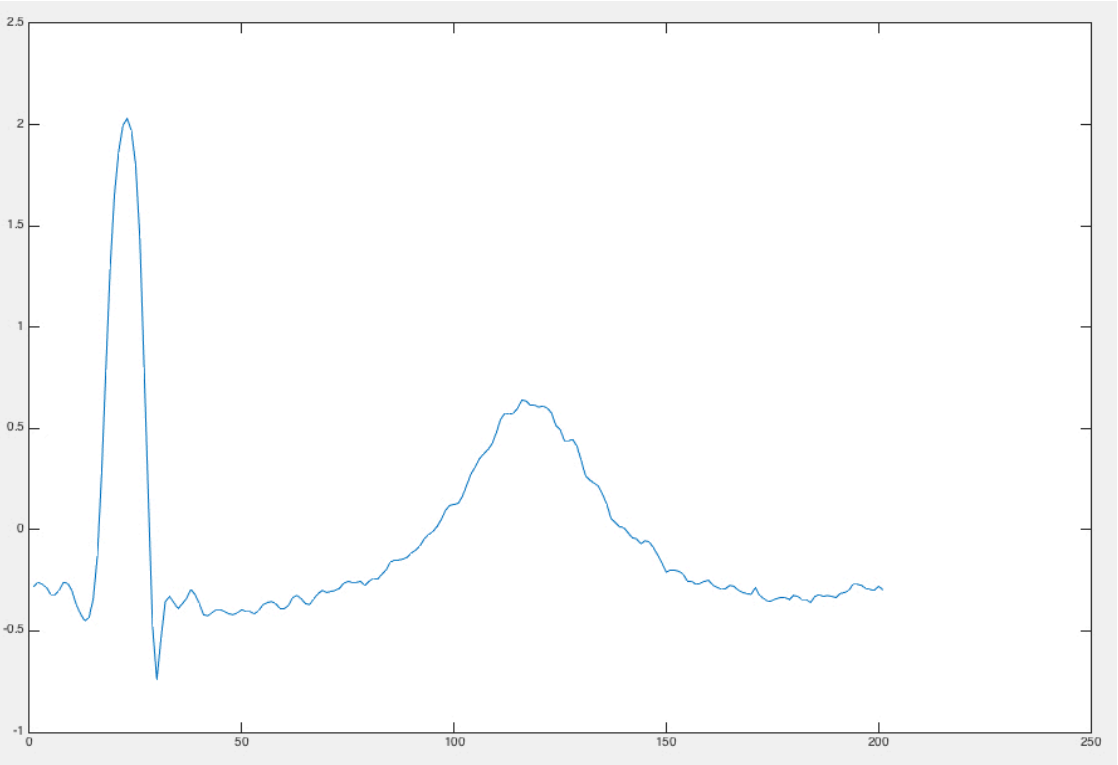


Figura 40. Latido original

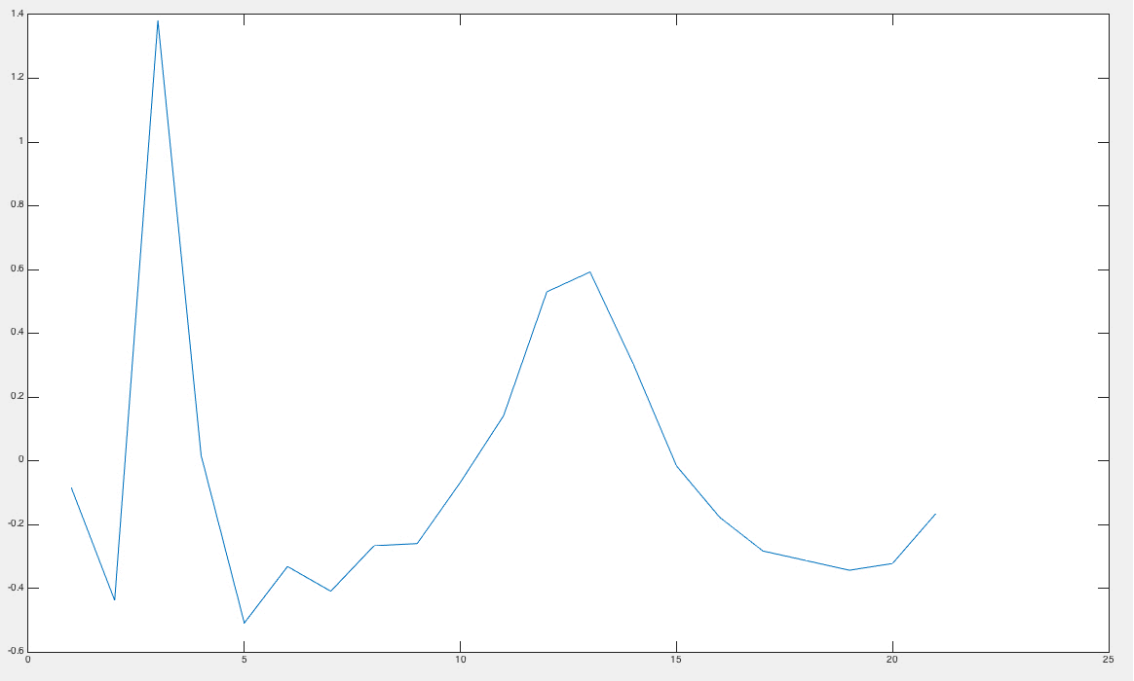
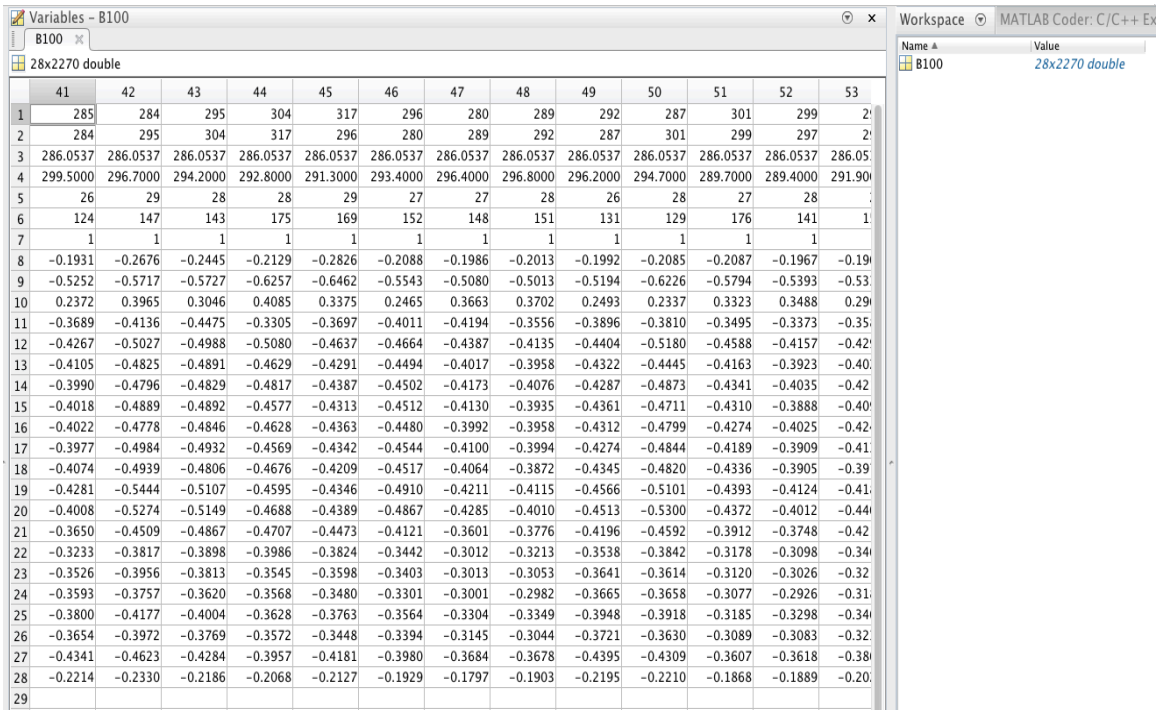


Figura 41. Latido remuestreado

6.4. Almacenamiento y estructura de los datos

En el apartado anterior hemos visto como se extraían de cada latido 28 características en total, 6 valores de duración de intervalos, 1 valor booleano de presencia de onda y 21 valores de morfología de la señal, que van a ser los datos de entrada a la Red Neuronal que se diseñara para clasificar los latidos. En primer lugar los datos se clasificaron registro a registro, es decir, por ejemplo en el registro 100 hay un total de 2270 latidos (en realidad el registro 100 está compuesto por 2272 latidos pero se tuvo que prescindir del primer y último latido del registro debido a que había algunos valores sobre los que era imposible tener información, como el intervalo pre RR del primer latido; no se puede calcular porque no hay latido anterior. Esta situación se repite en todos los registros que poseen dos latidos menos que los registros originales) con 28 características cada uno por lo que fueron almacenados en una matriz de Matlab de 28 filas x 2270 columnas como vemos en Figura 42. Las columnas corresponden al número de latido del registro y cada una de las filas corresponde a una característica de ese latido, ordenadas como se describen en el apartado anterior.

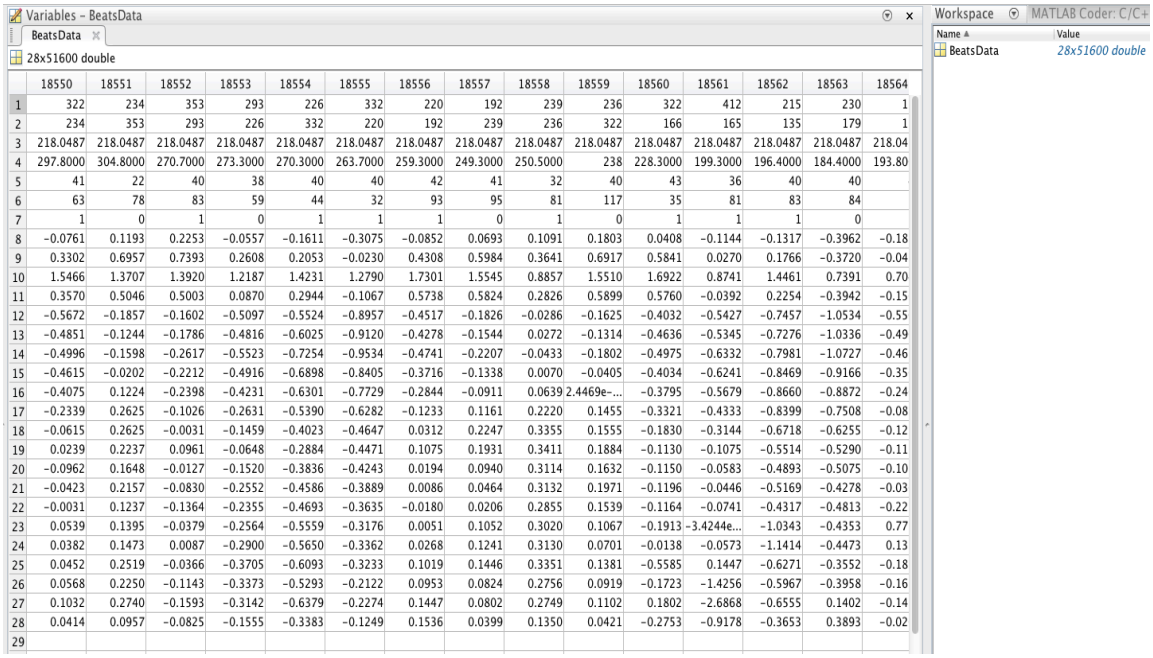


	41	42	43	44	45	46	47	48	49	50	51	52	53
1	285	284	295	304	317	296	280	289	292	287	301	299	2
2	284	295	304	317	296	280	289	292	287	301	299	297	2
3	286.0537	286.0537	286.0537	286.0537	286.0537	286.0537	286.0537	286.0537	286.0537	286.0537	286.0537	286.0537	286.05
4	299.5000	296.7000	294.2000	292.8000	291.3000	293.4000	296.4000	296.8000	296.2000	294.7000	289.7000	289.4000	291.90
5	26	29	28	28	29	27	27	28	26	28	27	28	
6	124	147	143	175	169	152	148	151	131	129	176	141	1
7	1	1	1	1	1	1	1	1	1	1	1	1	
8	-0.1931	-0.2676	-0.2445	-0.2129	-0.2826	-0.2088	-0.1986	-0.2013	-0.1992	-0.2085	-0.2087	-0.1967	-0.19
9	-0.5252	-0.5717	-0.5727	-0.6257	-0.6462	-0.5543	-0.5080	-0.5013	-0.5194	-0.6226	-0.5794	-0.5393	-0.53
10	0.2372	0.3965	0.3046	0.4085	0.3375	0.2465	0.3663	0.3702	0.2493	0.2337	0.3323	0.3488	0.29
11	-0.3689	-0.4136	-0.4475	-0.3305	-0.3697	-0.4011	-0.4194	-0.3556	-0.3896	-0.3810	-0.3495	-0.3373	-0.35
12	-0.4267	-0.5027	-0.4988	-0.5080	-0.4637	-0.4664	-0.4387	-0.4135	-0.4404	-0.5180	-0.4588	-0.4157	-0.42
13	-0.4105	-0.4825	-0.4891	-0.4629	-0.4291	-0.4494	-0.4017	-0.3958	-0.4322	-0.4445	-0.4163	-0.3923	-0.40
14	-0.3990	-0.4796	-0.4829	-0.4817	-0.4387	-0.4502	-0.4173	-0.4076	-0.4287	-0.4873	-0.4341	-0.4035	-0.42
15	-0.4018	-0.4889	-0.4892	-0.4577	-0.4313	-0.4512	-0.4130	-0.3935	-0.4361	-0.4711	-0.4310	-0.3888	-0.40
16	-0.4022	-0.4778	-0.4846	-0.4628	-0.4363	-0.4480	-0.3992	-0.3958	-0.4312	-0.4799	-0.4274	-0.4025	-0.42
17	-0.3977	-0.4984	-0.4932	-0.4569	-0.4342	-0.4544	-0.4100	-0.3994	-0.4274	-0.4844	-0.4189	-0.3909	-0.41
18	-0.4074	-0.4939	-0.4806	-0.4676	-0.4209	-0.4517	-0.4064	-0.3872	-0.4345	-0.4820	-0.4336	-0.3905	-0.39
19	-0.4281	-0.5444	-0.5107	-0.4595	-0.4346	-0.4910	-0.4211	-0.4115	-0.4566	-0.5101	-0.4393	-0.4124	-0.41
20	-0.4008	-0.5274	-0.5149	-0.4688	-0.4389	-0.4867	-0.4285	-0.4010	-0.4513	-0.5300	-0.4372	-0.4012	-0.44
21	-0.3650	-0.4509	-0.4867	-0.4707	-0.4473	-0.4121	-0.3601	-0.3776	-0.4196	-0.4592	-0.3912	-0.3748	-0.42
22	-0.3233	-0.3817	-0.3898	-0.3986	-0.3824	-0.3442	-0.3012	-0.3213	-0.3538	-0.3842	-0.3178	-0.3098	-0.34
23	-0.3526	-0.3956	-0.3813	-0.3545	-0.3598	-0.3403	-0.3013	-0.3053	-0.3641	-0.3614	-0.3120	-0.3026	-0.32
24	-0.3593	-0.3757	-0.3620	-0.3568	-0.3480	-0.3301	-0.3001	-0.2982	-0.3665	-0.3658	-0.3077	-0.2926	-0.31
25	-0.3800	-0.4177	-0.4004	-0.3628	-0.3763	-0.3564	-0.3304	-0.3349	-0.3948	-0.3918	-0.3185	-0.3298	-0.34
26	-0.3654	-0.3972	-0.3769	-0.3572	-0.3448	-0.3394	-0.3145	-0.3044	-0.3721	-0.3630	-0.3089	-0.3083	-0.32
27	-0.4341	-0.4623	-0.4284	-0.3957	-0.4181	-0.3980	-0.3684	-0.3678	-0.4395	-0.4309	-0.3607	-0.3618	-0.38
28	-0.2214	-0.2330	-0.2186	-0.2068	-0.2127	-0.1929	-0.1797	-0.1903	-0.2195	-0.2210	-0.1868	-0.1889	-0.20
29													

Figura 42. Matriz de características de latidos del registro 100 de MIT-BIH Arritmia

Una vez se han extraído las características de los 21 registros utilizados en el proyecto, se tienen guardadas en Matlab 21 matrices de 28 filas x (nº de latidos en cada registro) columnas. Por ejemplo el registro 109 contiene 2530 latidos o el registro 223 que contiene 2603 latidos. A continuación se va guardando cada una de las matrices de cada registro en una nueva matriz donde estarán todos los beats de todos los registros, es decir, una matriz de 28 características x

51600 latidos llamada *BeatsData,* que vemos en la Figura 43 , y que forma todos los datos que vamos a utilizar en este proyecto.



Variables - BeatsData

BeatsData

28x51600 double

	18550	18551	18552	18553	18554	18555	18556	18557	18558	18559	18560	18561	18562	18563	18564
1	322	234	353	293	226	332	220	192	239	236	322	412	215	230	1
2	234	353	293	226	332	220	192	239	236	322	166	165	135	179	1
3	218.0487	218.0487	218.0487	218.0487	218.0487	218.0487	218.0487	218.0487	218.0487	218.0487	218.0487	218.0487	218.0487	218.0487	218.04
4	297.8000	304.8000	270.7000	273.3000	270.3000	263.7000	259.3000	249.3000	250.5000	238	228.3000	199.3000	196.4000	184.4000	193.80
5	41	22	40	38	40	40	42	41	32	40	43	36	40	40	
6	63	78	83	59	44	32	93	95	81	117	35	81	83	84	
7	1	0	1	0	1	1	1	0	1	0	1	1	1	0	
8	-0.0761	0.1193	0.2253	-0.0557	-0.1611	-0.3075	-0.0852	0.0693	0.1091	0.1803	0.0408	-0.1144	-0.1317	-0.3962	-0.18
9	0.3302	0.6957	0.7393	0.2608	0.2053	-0.0230	0.4308	0.5984	0.3641	0.6917	0.5841	0.0270	0.1766	-0.3720	-0.04
10	1.5466	1.3707	1.3920	1.2187	1.4231	1.2790	1.7301	1.5545	0.8857	1.5510	1.6922	0.8741	1.4461	0.7391	0.70
11	0.3570	0.5046	0.5003	0.0870	0.2944	-0.1067	0.5738	0.5824	0.2826	0.5899	0.5760	-0.0392	0.2254	-0.3942	-0.15
12	-0.5672	-0.1857	-0.1602	-0.5097	-0.5524	-0.8957	-0.4517	-0.1826	-0.0286	-0.1625	-0.4032	-0.5427	-0.7457	-1.0534	-0.55
13	-0.4851	-0.1244	-0.1786	-0.4816	-0.6025	-0.9120	-0.4278	-0.1544	0.0272	-0.1314	-0.4636	-0.5345	-0.7276	-1.0336	-0.49
14	-0.4996	-0.1598	-0.2617	-0.5523	-0.7254	-0.9534	-0.4741	-0.2207	-0.0433	-0.1802	-0.4975	-0.6332	-0.7981	-1.0727	-0.46
15	-0.4615	-0.0202	-0.2212	-0.4916	-0.6898	-0.8405	-0.3716	-0.1338	0.0070	-0.0405	-0.4034	-0.6241	-0.8469	-0.9166	-0.35
16	-0.4075	0.1224	-0.2398	-0.4231	-0.6301	-0.7729	-0.2844	-0.0911	0.0639	2.4469e-...	-0.3795	-0.5679	-0.8660	-0.8872	-0.24
17	-0.2339	0.2625	-0.1026	-0.2631	-0.5390	-0.6282	-0.1233	0.1161	0.2220	0.1455	-0.3321	-0.4333	-0.8399	-0.7508	-0.08
18	-0.0615	0.2625	-0.0031	-0.1459	-0.4023	-0.4647	0.0312	0.2247	0.3355	0.1555	-0.1830	-0.3144	-0.6718	-0.6255	-0.12
19	0.0239	0.2237	0.0961	-0.0648	-0.2884	-0.4471	0.1075	0.1931	0.3411	0.1884	-0.1130	-0.1075	-0.5514	-0.5290	-0.11
20	-0.0962	0.1648	-0.0127	-0.1520	-0.3836	-0.4243	0.0194	0.0940	0.3114	0.1632	-0.1150	-0.0583	-0.4893	-0.5075	-0.10
21	-0.0423	0.2157	-0.0830	-0.2552	-0.4586	-0.3889	0.0086	0.0464	0.3132	0.1971	-0.1196	-0.0446	-0.5169	-0.4278	-0.03
22	-0.0031	0.1237	-0.1364	-0.2355	-0.4693	-0.3635	-0.0180	0.0206	0.2855	0.1539	-0.1164	-0.0741	-0.4317	-0.4813	-0.22
23	0.0539	0.1395	-0.0379	-0.2564	-0.5559	-0.3176	0.0051	0.1052	0.3020	0.1067	-0.1913	-3.4244e...	-1.0343	-0.4353	0.77
24	0.0382	0.1473	0.0087	-0.2900	-0.5650	-0.3362	0.0268	0.1241	0.3130	0.0701	-0.0138	-0.0573	-1.1414	-0.4473	0.13
25	0.0452	0.2519	-0.0366	-0.3705	-0.6093	-0.3233	0.1019	0.1446	0.3351	0.1381	-0.5585	0.1447	-0.6271	-0.3552	-0.18
26	0.0568	0.2250	-0.1143	-0.3373	-0.5293	-0.2122	0.0953	0.0824	0.2756	0.0919	-0.1723	-1.4256	-0.5967	-0.3958	-0.16
27	0.1032	0.2740	-0.1593	-0.3142	-0.6379	-0.2274	0.1447	0.0802	0.2749	0.1102	0.1802	-2.6868	-0.6555	0.1402	-0.14
28	0.0414	0.0957	-0.0825	-0.1555	-0.3383	-0.1249	0.1536	0.0399	0.1350	0.0421	-0.2753	-0.9178	-0.3653	0.3893	-0.02
29															

Workspace MATLAB Coder: C/C+

Name	Value
BeatsData	28x51600 double

Figura 43. Matriz que tiene todos los datos usados en el proyecto, *BeatsData* (28 x 51600)

Ahora que tenemos todos los valores y características de todos los latidos de los 21 registros podemos saber los valores medios de cada uno de los 5 tipos de latidos de la ANSI/AAMI EC57:2012 [19]. Realmente no sería necesario, puesto que la Red Neuronal es capaz de clasificar y aprender sin necesidad de tener como entrada los valores típicos de cada tipo de latido, ella misma los reconoce con el aprendizaje, pero es de utilidad para la comprensión del proyecto. Para clasificar los beats por tipo se crearon 5 nuevas matrices llamadas *Nbeats, Sbeats, Vbeats, Fbeats y Qbeats* cada una de las cuales contiene los distintos tipos de latido. Para crear estas matrices a la matriz global de todos los latidos, *BeatsData,* se le añadió una nueva fila superior (que se usa exclusivamente para esta tarea, después es borrada) que tiene el código ASCII (Matlab no permite en una matriz numérica incluir letras) de cada uno de los tipos de latido, es decir, si un latido es de tipo Normal llevara en a fila superior un '78' equivalente a su etiqueta N, y así para cada uno de los cuatro restantes tipos de latido; este número permite identificar el tipo de latido para separarlo y guardarlo en su matriz correspondiente. La conversión de números a caracteres fue realizada en Excel con la función CODIGO.

Por lo tanto una vez tenemos estas cinco nuevas matrices (*Nbeats, Sbeats, Vbeats, Fbeats y Qbeats)* que contienen los latidos de cada tipo se pueden calcular los valores típicos de cada clase de latido y sacar conclusiones. En la siguiente Tabla 6 podemos observar los valores típicos de cada una de las característica de los diferentes tipos de latido:

Tipo de latido ANSI / AAMI	N	S	V	F	Q
Pre RR	276,06	223,42	183,40	206,82	359,60
Post RR	256,06	312,32	302,25	190,09	325,67
Complejo QRS	33,35	30,91	46,80	39,03	52,33
Intervalo ST	110,93	103,65	93,65	97,74	79,80
Valor booleano onda P	0,90	0,72	0,38	0,95	0,6

Tabla 6. Valores medios de cada característica para cada tipo de latido ANSI / AAMI

Para comprender la Tabla 6 lo primero que hay que explicar son las unidades en las que están expresadas las características. Las cuatro primeras características son duraciones de ondas, por lo que debería ser unidades de tiempo, pero están expresadas en nº de muestras, esto quiere decir, nuestros registros son de 30:05 minutos a 360 Hz por lo que tenemos 650000 muestras por registro. Nuestro periodo a 360 Hz es de 2,$\hat{7}$ ms, por lo que cada muestra representa un tiempo de 2,$\hat{7}$ ms. Para la comprensión de la tabla si por ejemplo tenemos un valor de 300, en tiempo serían 300 muestras x 2,$\hat{7}$ ms = 0,83 segundos. La ultima características es un valor booleano y por tanto si el valor se acerca mucho a 1 quiere decir que en ese tipo de latido casi todos los latidos tienen onda P, y si se acerca mucho a 0 quiere decir que casi ninguno tiene onda P. Los valores no han sido convertidos a tiempo puesto que la Red Neuronal no necesita que los datos este en unidades de tiempo.

Analizando la tabla podemos ver que, por ejemplo, para los tipos S y V los valores pre RR suelen ser más cortos de lo normal y, en cambio, los valores post RR suelen ser más largos de lo normal. El complejo QRS más ancho lo tienen los latidos V, casi todo los latidos F tienen onda P, etc. Estas mismas conclusiones que estamos extrayendo de la tabla, la Red Neuronal que vamos a diseñar las extrae ella misma para poder clasificar los latidos, en cambio si podría valer para utilizar otro tipo de clasificadores.

6.5. Diseño de la Red Neuronal para clasificar latidos

El diseño y la elección de la Red Neuronal, así como del número de neuronas en las capas ocultas, es uno de los mayores problemas a los que se puede enfrentar quien quiera resolver un problema con la aplicación de una Red Neuronal, porque no existe ninguna regla generalizadas sino que el diseñador deberá utilizar el método de prueba y error para lograr una red que funcione correctamente para cada aplicación. Las dos primeras características del diseño nos las dan los datos de entrada a la red. Como se ha explicado anteriormente, cada uno de nuestros latidos tiene 28 características que serán los datos de entrada a la Red Neuronal y tendremos una salida por cada dato.

En primer lugar se diseño en Matlab una Red Neuronal de tipo Perceptron Multicapa con dos capas ocultas de 30 y 25 neuronas cada una, y con una salida de tipo entero para clasificar, es decir, los tipos de latido a la salida de la red habían sido codificados para leer la respuesta de la Red Neuronal de esta manera: Latido Normal = 1, Latido Ectópico Supraventricular = 2, Latido Ectópico Ventricular = 3, Latido Fusionado = 4 y Latido Desconocido = 5. Se realizaron y probaron entrenamientos con distintas condiciones y algoritmos, pero el resultado no fue satisfactorio, la red no era capaz de generalizar correctamente. Además fue evidente comprobar que a una Red de este tipo le es más complicado y lento aprender a mostrar las salidas en forma de número entero codificado para cada tipo (como se había hecho en este proyecto: 1, 2, 3, 4, 5) que hacerlo en forma de vector con 1 y 0.

Tras este primer diseño fallido se opto por cambiar de estructura de Red Neuronal y se uso una red especializada en aplicaciones de clasificación, una Red de Reconocimiento de Patrones (Figura 44) (*Pattern Recognition Network*). Este tipo de redes son redes *feedforward* (como se explico en el capítulo 3, son redes con conexiones solo en una dirección y con aprendizaje supervisado) con dos capas ocultas, función de activación sigmoide y neuronas de salida de tipo Softmax, que es la capa oculta que permite el reconocimiento de clases. Pueden ser entrenadas para clasificar las entradas de acuerdo a las clases o tipos de salida. Los datos de salida de este tipo de redes deben consistir en vectores de todos los valores a cero, excepto un 1 en un elemento del vector, que indicara a que tipo de la salida pertenece la entrada. Con esto, esto llegamos a la misma conclusión que cuando diseñamos el Perceptron Multicapa anteriormente, las salidas deben estar codificadas en forma de vectores en aplicaciones de clasificación para una mayor eficiencia de la red. Por lo tanto una vez tenemos decidido el diseño de la Red Neuronal se procederá a su entrenamiento con diferente número de neuronas en la primera capa oculta y diferentes algoritmos de entrenamiento.

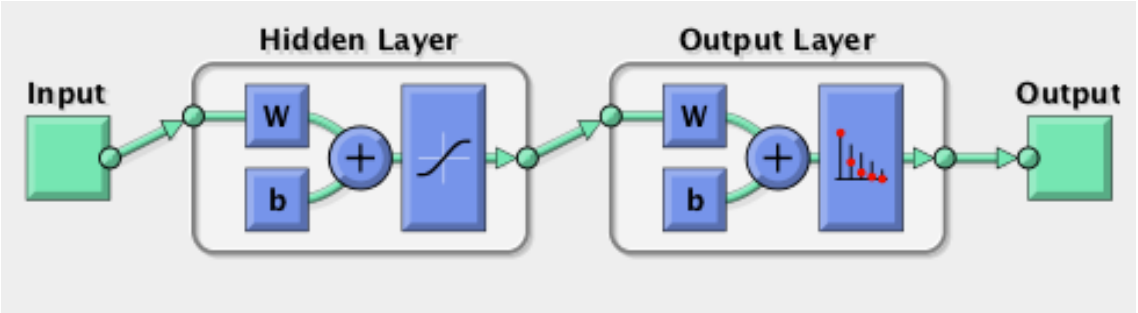


Figura 44. Estructura de una Red de Reconocimiento de Patrones

6.6. Entrenamiento y validación de la Red Neuronal

Una vez elegida la estructura de la Red utilizamos la aplicación de Matlab para redes neuronales, *Neural Network Toolbox* [32], para crear y entrenar la red. Lo primero de todo es separar los datos en tres conjuntos: entrenamiento, validación y test:

- ***Datos de entrenamiento***: Son los datos utilizados para entrenar la red y ajustar los pesos para dar una respuesta correcta.
- ***Datos de validación***: Estos datos son utilizados durante el entrenamiento para comprobar el proceso de entrenamiento. Una de las condiciones de detención del entrenamiento puede ser un número determinado de comprobaciones, en las que aplicando el conjunto de validación el error no disminuye, por lo tanto el entrenamiento ha concluido porque el error no va a bajar más.
- ***Datos de test***: Estos datos son utilizados para validar y comprobar el correcto funcionamiento de la red neuronal cuando el entrenamiento ha finalizado, cuando los pesos ya están fijos.

Los datos de entrenamiento (un total de 51600 latidos con 28 características cada uno) se dividen aleatoriamente en estos tres conjuntos con los siguientes porcentajes: entrenamiento (70%, 36120 latidos), validación (15%, 7740 latidos) y test (15%, 7740 latidos). Si el conjunto de datos que tenemos son significativos y representativos (como hemos demostrado en apartados anteriores) la mejor opción es dividir los conjuntos aleatoriamente, y para las aplicaciones con una alta cantidad de datos se recomienda dividirlos en 70% - 15% - 15% como hemos realizado. Al final del proceso, al evaluar el resultado de la red, podremos darnos cuenta si la separación de datos ha sido correcta. Así mismo se creo una matriz de 5 x 51600, necesaria para entrenar la Red con aprendizaje supervisado, y en la que en cada columna todos los valores son 0 excepto una posición donde haya 1 y que será la clase a la que pertenece ese latido en nuestro caso. También hay que añadir que no es necesario normalizar los datos manualmente, puesto que Matlab aplica automáticamente una función de normalización a todos los datos de entrada para una optimización de la Red Neuronal.

El siguiente paso es elegir el número de neuronas en la capa oculta y elegir el algoritmo de entrenamiento. Se utilizan y prueban varios tipos de algoritmos de entrenamiento recomendados para este tipo de aplicaciones [33] como el algoritmo de *Levenberg-Marqueardt*, el de *Polark-Ribiére Conjugate Gradient*, utilizando finalmente para entrenar nuestra red el algoritmo *Scaled Conjugate Gradiente* (trainscg), que aparte de ser el recomendado para aplicaciones de reconocimiento de patrones y muchos datos de entrenamiento, se comprueba experimentalmente que es el que más rápido converge. Una vez tenemos

elegido el algoritmo se realizaron varios entrenamientos con distintos números de neuronas en la primera capa para ver cual era la mejor opción (el número de neuronas en la segunda capa viene dado por el vector salida de la red en nuestro caso un vector de 5 filas x 1 columna para cada latido, por lo que en esa capa habrá 5 neuronas de clasificación). Los resultados de los entrenamientos para distinto número de neuronas en la primera capa los vemos en la siguiente Tabla 7:

Nº Neuronas capa oculta	Error	% de aciertos en N	% de aciertos en S	% de aciertos en V	% de aciertos en F	% de aciertos en Q	% de aciertos total
20	0,0284	99,3%	85,1%	94,1%	70,4%	0%	97,6%
30	0,0276	99,4%	86,4%	94,2%	71,2%	0%	97,8%
35	***0,0217***	***99,4%***	***88,9%***	***95,3%***	***74,7%***	***6,7%***	***98,1%***
40	0,275	99,4%	84,7%	94,3%	67,2%	6,7%	97,7%
50	0,0297	99,4%	82,6%	94,3%	68%	0%	97,6%

Tabla 7. Resultados del entrenamiento con diferente nº de neuronas en la capa oculta

Como se puede observar en la Tabla 7, el menor error y el mayor porcentaje de aciertos al clasificar los latidos se obtiene con 35 neuronas en la capa oculta, por lo que elegiremos este diseño para nuestra Red. Del resto de la tabla podemos sacar las conclusiones de que los tipos de latido que mejor clasificados son las clase N, S, y V, al ser las clases de las que mayo número de ejemplos se dispone, sobretodo la N. En los latidos de tipo Q el porcentaje disminuye al 6,7% debido a que se necesitarían más ejemplos de este tipo pero la base de datos no los posee, al ser muy raramente observables en ECGs. Por lo tanto, después de todas las pruebas para comprobar cual es el diseño de Red que mejores resultados da, finalmente nuestra Red tendrá 35 neuronas en la capa oculta y será de este tipo como la Figura 45:

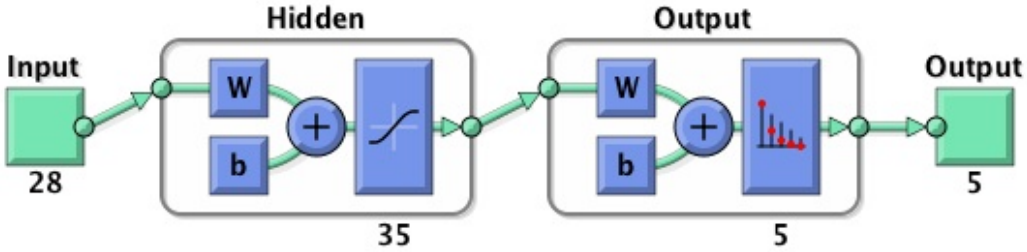


Figura 45. Diseño final de la Red Neuronal utilizada.

Entrenamos la Red con el algoritmo *Scaled Conjugate Gradiente* (trainscg), y estos son los resultados obtenidos en el entrenamiento:

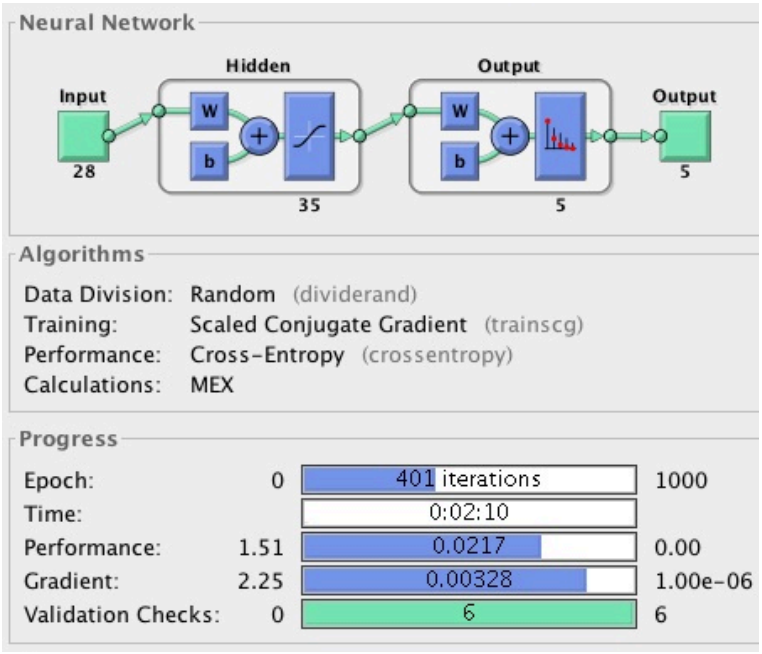


Figura 46. Resultados del entrenamiento de la Red

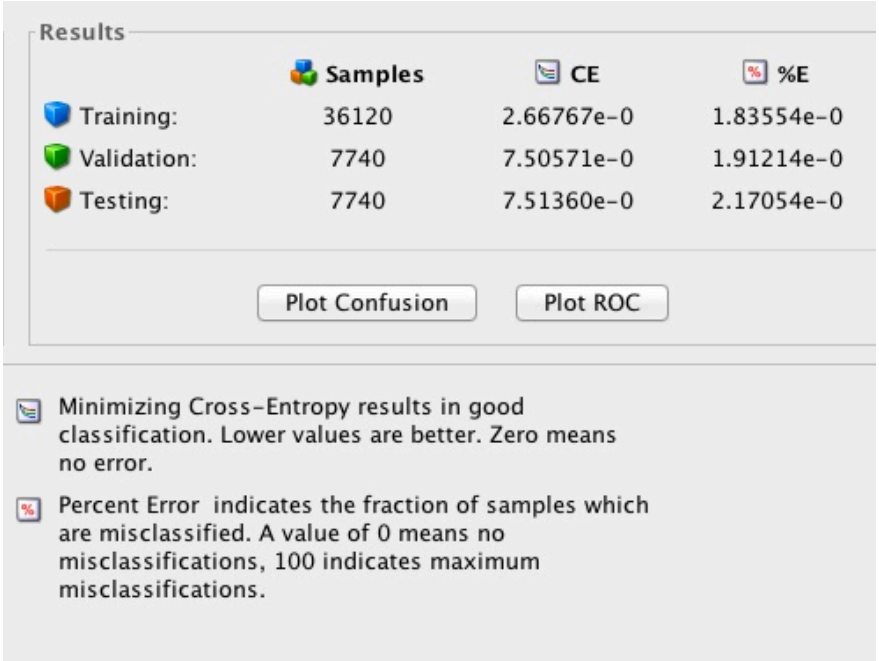


Figura 47. Resultados del entrenamiento de la Red (2)

Como podemos observar en la Figura 46 y Figura 47, los resultados son satisfactorios puesto que la tasa de error baja hasta el 0,0217 y los valores de entropía cruzada (método para la estimación de probabilidades) y porcentaje de error son aceptables. En las siguientes Figura 48 y Figura 49, se pueden ver los resultados en la matriz de confusión, que indica los porcentajes de acierto y error de clasificación para cada tipo de latido en cada uno de los conjuntos de datos.

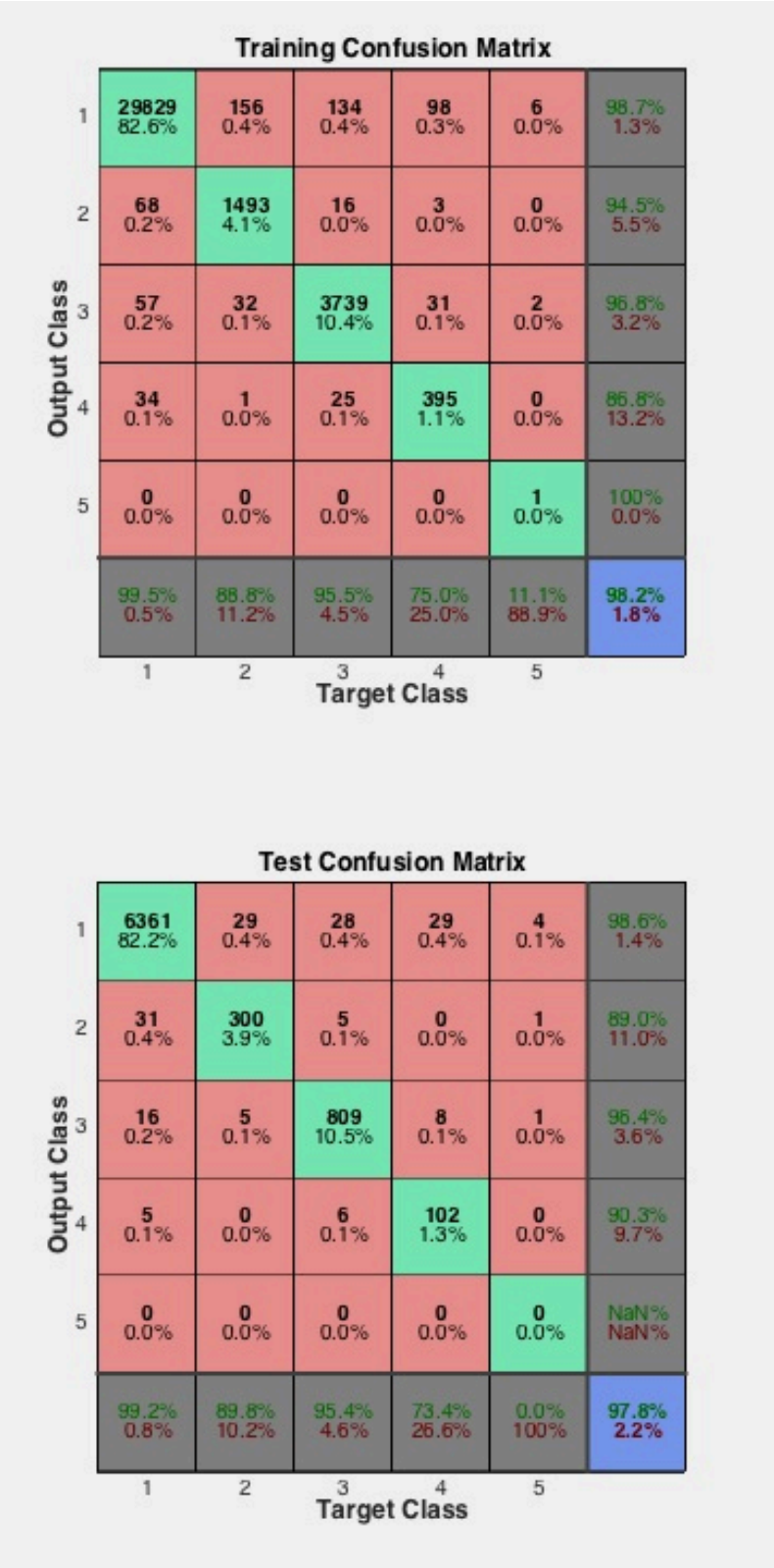


Figura 48. Matriz de confusión para los datos de entrenamiento y test

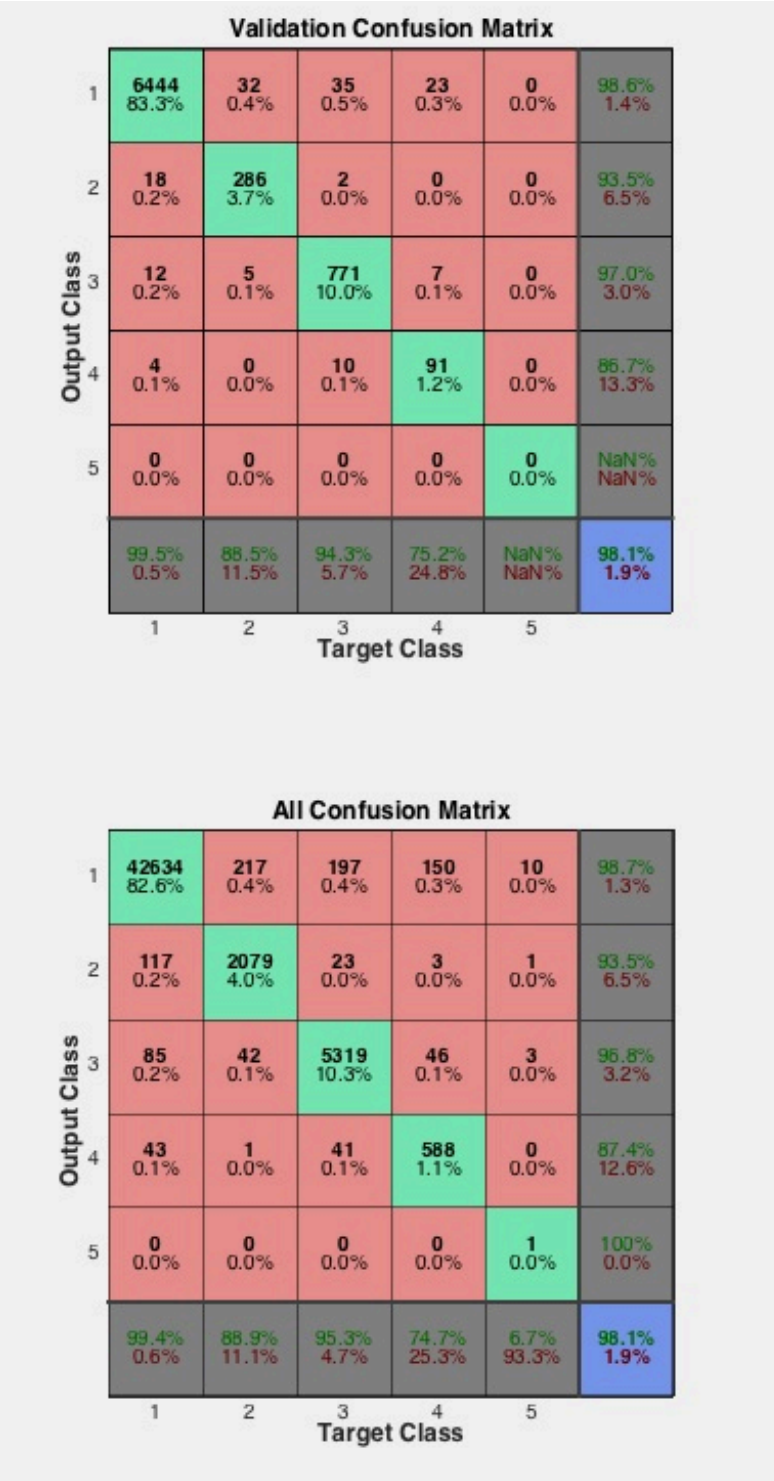


Figura 49. Matriz de confusión para los datos de validación y datos totales

En las dos figuras anteriores se pueden ver los resultados en la matriz de confusión para los datos de entrenamiento, test, validación y para todos los datos. Se pueden ver los resultados para cada tipo de latido tanto a la entrada como a la salida. Como hemos explicado anteriormente las cuatro primeras clases (N, S, V y F) tienen altos porcentajes de clasificación correctos (99,4%, 88,9%, 95,3% y 74,7% respectivamente) pero la clase Q con solo un 6,7% no es clasificada correctamente, por lo que se ha comentado en párrafos

anteriores en la Tabla 7. Analizando la matriz de confusión referida al total de los datos utilizados, llegamos a la conclusión de que se ha clasificado bien **50621** latidos de un total de **51600**, y hay **979** latidos mal clasificados, es decir, la Red diseñada clasifica bien el **98,1%** de los latidos, lo que en principio parecen unos resultados satisfactorios, excepto para la clase Q. A continuación compararemos los resultados con los resultados obtenidos por otros métodos de clasificación de latidos.

Observando las matrices de confusión y los porcentajes de clasificación correctos de los trabajos mencionados en este proyecto, estos son algunos resultados; tenemos un 93,88% de latidos clasificados correctamente en [24], un 92,4% en [34], un 85,87% en [23] o un 86,4% en [25]. Además podemos encontrar más resultados de aplicaciones clasificación en [25] donde hace una comparación con otros métodos publicados donde destacan artículos como [35], [36] o [37]. A la vista de los datos y las investigaciones realizadas hasta la fecha el porcentaje de latidos correctamente clasificados en este proyecto (98,1%) es un valor de aciertos satisfactorio que permitiría seguir desarrollando el proyecto para su optimización. Es importante tener en cuenta que la mayoría de los artículos citados tienen el mismo problema en la clasificación de los latidos de tipo Q, debido a que se ha utilizado la misma base de datos la MIT-BIH Arritmia [17], y esta carece de suficientes ejemplos de ese tipo de latido. Por último señalar que todos estos trabajos citados excluyen, como también se ha hecho en este trabajo, los cuatro registros (102, 104, 107 y 217) de ritmo acelerado, como recomienda la AAMI [26] y como se ha explicado en apartados anteriores. Por lo tanto podemos concluir que la Red Neuronal que clasifica latidos según los tipos de la AAMI [19] ha sido correctamente diseñada y validada en Matlab, y se puede implementar en la plataforma Intel Edison, como se explicara en el capítulo siguiente.

6.7. Algoritmo Pan-Tompkins en Matlab

Uno de los objetivos del proyecto es utilizar el algoritmo de detección de complejos QRS de Pan-Tompkins [21], [22], para calcular el ritmo cardíaco. Para implementar el algoritmo en Matlab hemos utilizado *"Complete Pan Tomkins Implementation ECG QRS Detector"* [38] distribuido en Mathworks. Esta implementación consta de un archivo .m donde está desarrollado el código de detección y umbralización adaptativa, con todas sus etapas como está descrito en el capítulo 5.

Una vez tenemos el algoritmo implementado, pasamos a la selección de datos. Como explicamos en apartados anteriores, se decidieron usar 25 registros de la base de datos MIT-BIH Arritmia [17] de 10 segundos (a 360 Hz, 3600 muestras) de duración cada uno para posteriormente calcular el ritmo cardíaco. Los registros completos fueron cargados en Matlab mediante la función **load** y guardados en una matriz 3600 valores de cada registro para tener un registro de 10 segundos de duración. A continuación (Figura 50 y Figura 51) se muestran los resultados de ejecutar el algoritmo para los 10 primeros segundos de la señal 100 del MIT-BIH Arritmia [17].

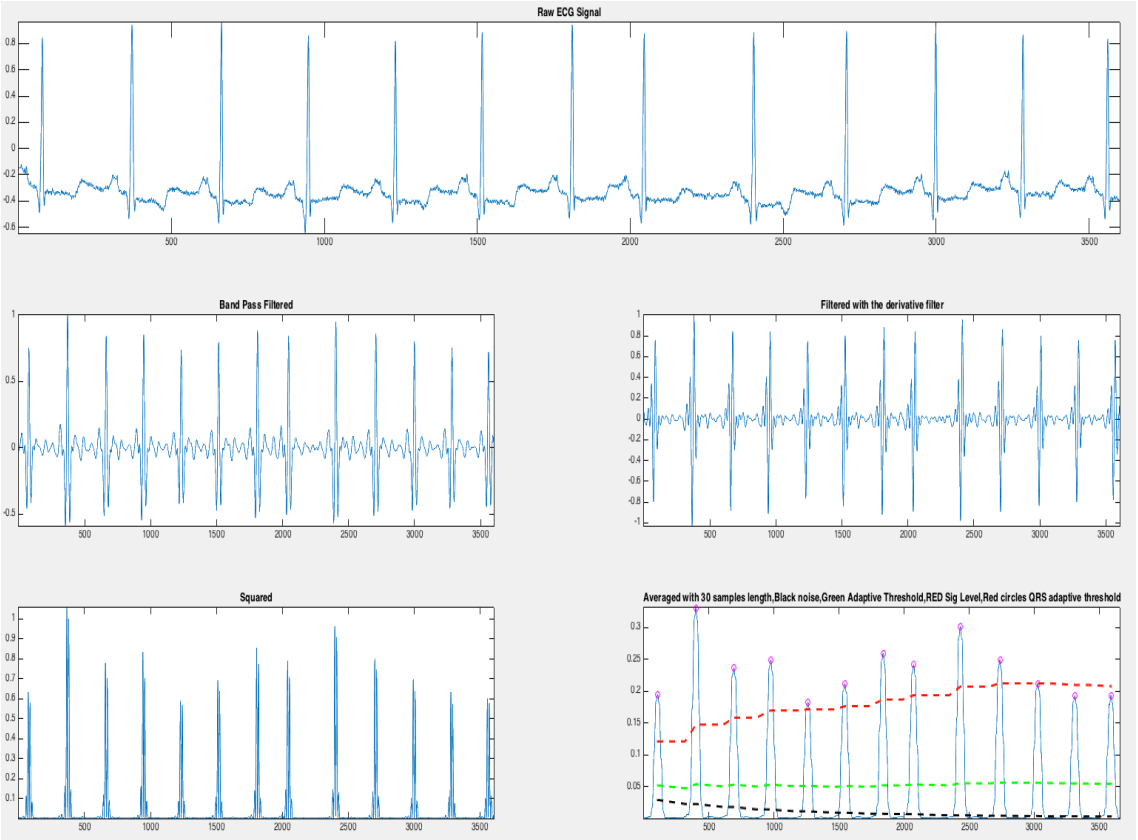


Figura 50. Señal original, señal filtrada, señal derivada, señal elevada al cuadrado y detección de QRS con algoritmo Pan-Tompkins en Matlab

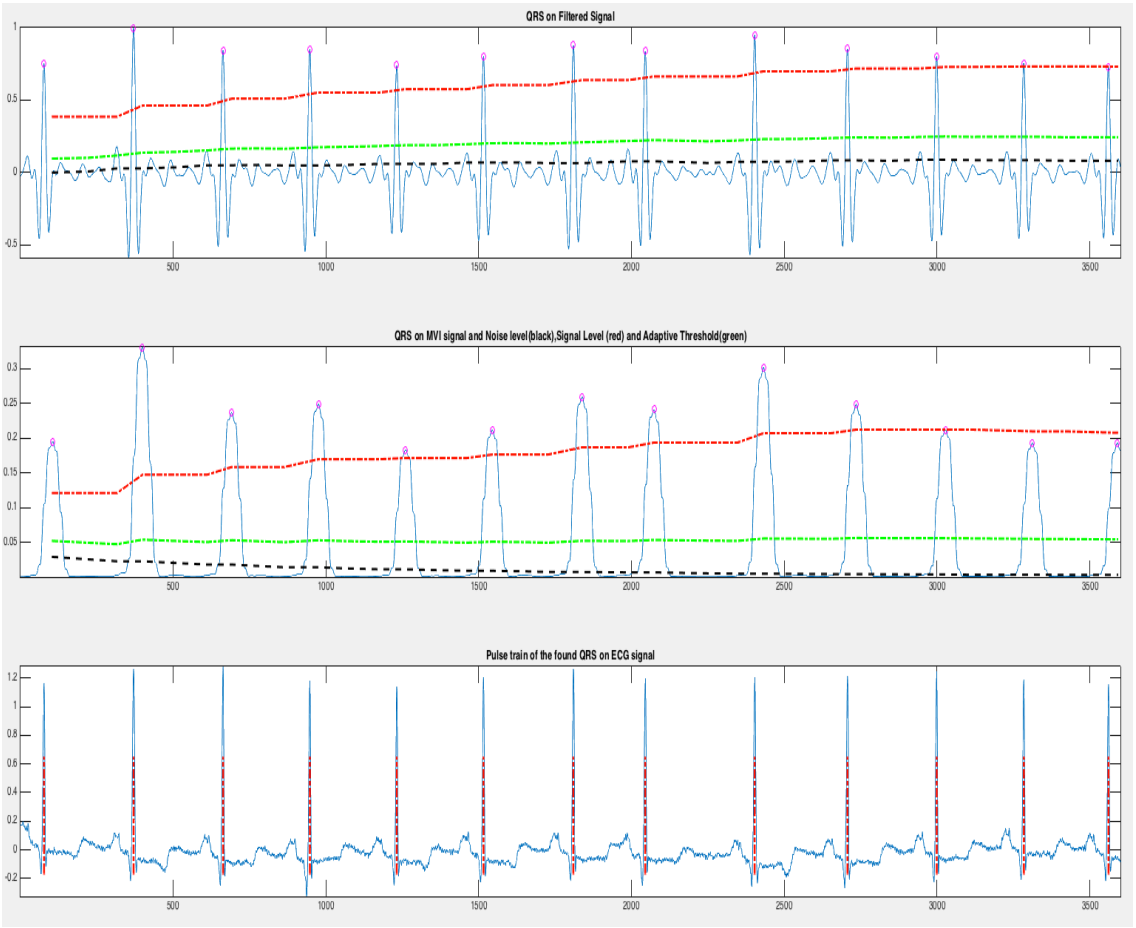


Figura 51. Detección de QRS con algoritmo Pan-Tompkins en Matlab

Estas figuras mostradas son los resultados gráficos de Matlab, pero la función utilizada **pan_tompkin** [38] tiene como resultados dos vectores: uno contiene el número de muestra donde está localizado el complejo QRS y el segundo vector contiene la amplitud de ese complejo QRS. En nuestro caso para el cálculo del ritmo cardíaco solo necesitamos la información de cuán largo es el vector que es lo que nos dará el número de latidos en esos 10 segundos. Podemos ver un ejemplo en la siguiente Figura 52:

qrs_amp_raw	*1x13 double*
qrs_i_raw	*1x13 double*

Figura 52. Vectores de salida de la función pan_tompkin [38]

Como podemos ver en la imagen tenemos dos vectores resultado, qrs_amp_raw es el vector donde se encuentran las amplitudes de cada complejo QRS detectado y qrs_i_raw es el vector donde se guarda el número de muestra de cada complejo QRS. Como hemos dicho para calcular el ritmo cardíaco solo nos interesa el largo del vector, en este caso para el registro 100, es de 13 lo que nos indica que en esos 10 segundos hay 13 latidos. Comprobamos que este dato es correcto haciendo uso de las herramientas de visualización de señales disponibles en la web de *Physionet* [3].

El siguiente paso es crear un programa en Matlab que reciba como entrada uno de los dos vectores salida de la función **pan_tompkin** [38], calcule el largo del vector, y con ese dato calcule el ritmo cardíaco por minuto. Este programa ha sido nombrado **HeartRatePanT.** Se comprueba que para todos los 25 registros de 10 segundos la función **pan_tompkin** [38] aplica correctamente el algoritmo y después la función **HeartRatePanT** calcula correctamente el ritmo cardíaco, siendo los resultados plenamente satisfactorios.

6.8. Matlab Coder

El último paso antes de implementar nuestros tres programas (la Red Neuronal diseñada, el algoritmo Pan-Tompkins y la función que calcula el ritmo cardíaco) en la plataforma Intel Edison es adaptarlos a código C, puesto que en la plataforma usaremos el entorno de desarrollo (IDE) Arduino [9]. Para ello utilizaremos una de las aplicaciones con las que cuenta Matlab, se trata de Matlab Coder [39], una aplicación que genera código C portátil. Es compatible con la mayor parte del lenguaje Matlab, y en nuestro caso se comprueba que las funciones a convertir son compatibles con la aplicación, excepto la función del algoritmo Pan_Tompkins. Para ello en vez de utilizar la funciono original, se utiliza "*Convert Pan-Tompkins Algorithm to C Using Matlab Coder*" [40] que funciona exactamente igual que la función **pan_tompkin** [38] explicada anteriormente, pero eliminando aquellas partes de la función no compatibles con Matlab Coder [39], que en este caso eran simplemente las funciones de representación en gráficas. Una vez convertidos los tres programas a código C solo queda implementarlos en la plataforma Intel Edison.

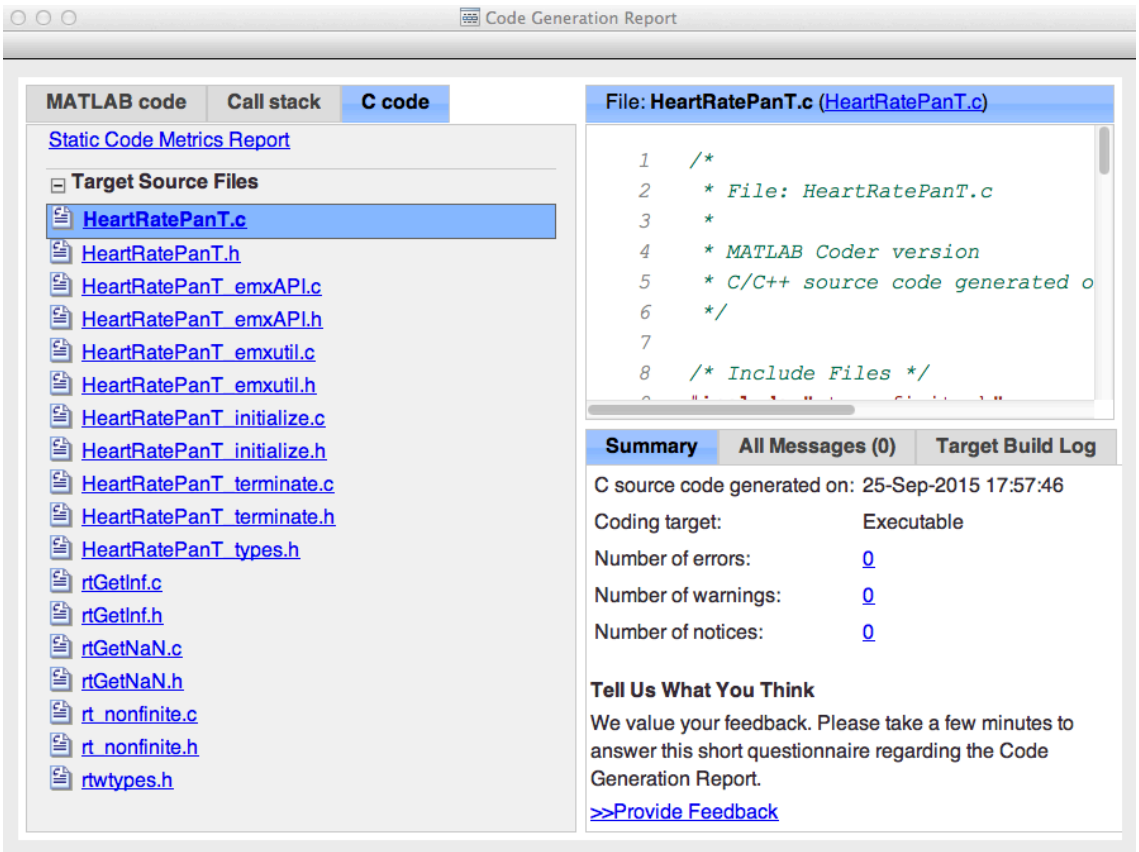


Figura 53. Informe de conversión a código C con Matlab Coder de la función HeartRatePanT

Capítulo 7

Implementación en plataforma Intel Edison

En este capítulo se explicará como se han implementado y ejecutado las funciones diseñadas en Matlab y convertidas a código C por Matlab Coder, en la plataforma Intel Edison.

7.1. Implementación y ejecución de los programas

Uno de los objetivos del proyecto, comentados en el capítulo 1, es comprobar que la plataforma Intel Edison [7], [8] tiene la capacidad de procesamiento y la memoria suficiente para desarrollar un sistema futuro que permita adquirir datos de un ECG para clasificar los latidos y calcular el ritmo cardíaco de un paciente y mostrar los resultados por pantalla. Para ello es necesario probar que la Intel Edison es capaz de ejecutar los programas de este proyecto, que serían la base para el sistema futuro propuesto, y que tiene capacidad suficiente para guardar señales ECG digitalizadas en su memoria.

El primer paso es guardar los archivos de registros ECG de la base de datos MIT-BIH Arritmia [17] en la plataforma. Aprovechando que la plataforma Intel Edison dispone de conexión Wifi, descargaremos los archivos de esta manera. Es necesario configurar manualmente la conexión Wifi siguiendo el manual [41]. Una vez la plataforma está conectada puede descargar datos vía Wifi, a través del protocolo de comunicación SSH, del ordenador; para ello utilizamos el comando **scp** en el terminal del ordenador indicando los archivos que queremos descargar. Es importante aclarar en este punto, que los archivos descargados en la plataforma deben ser .dat y nuestras matrices, con las características de los beat que funcionan como entrada a la Red Neuronal así como los registros de 10 s de duración que sirven como entrada al algoritmo Pan-Tompkin, están guardadas en .mat de Matlab. Transformamos los archivos con la función **dlmwrite** [42] de Matlab, de .mat a .dat y ya están preparados para descargarse y guardarse en la memoria de la Intel Edison.

A continuación explicaremos las etapas realizadas para ejecutar por una parte la Red Neuronal y por otra parte el cálculo del ritmo cardíaco con el algoritmo Pan-Tompkin.

- ***Clasificador de latidos con Red Neuronal:*** Con los registros ya guardados en la plataforma, el siguiente paso es desarrollar un programa en el entorno de desarrollo Arduino [9], explicado en el capítulo 2 y que será donde se ejecute la Red Neuronal, que permita cargar el registro que se quiera utilizar en un array multidimensional que será la entrada a la Red Neuronal. La Red Neuronal está diseñada para tener un número desconocido de entradas, por lo que podemos introducir un registro de 2400 latidos u otro de 4500, el único requisito es que tenga 28 columnas por latido, es decir, características. Usamos reserva de memoria dinámica para crear los arrays, y las funciones descritas en Matlab Coder para manejar estos arrays [43]. Una vez creados los arrays cargamos en ellos un registro que queramos procesar y ejecutamos, el programa irá mostrando por pantalla el tipo de latido como resultado. Para ello simplemente evaluamos el vector salida (de 5 posiciones) y busca la posición donde está el máximo (el valor que más se acerque a 1). Si el máximo está en la posición 0 será un Latido Normal N, si está en la posición 1 será un Latido Ectópico

Supraventricular, y así sucesivamente. Podemos ver un ejemplo del resultado por pantalla de la Red Neuronal en la siguiente Figura 54:

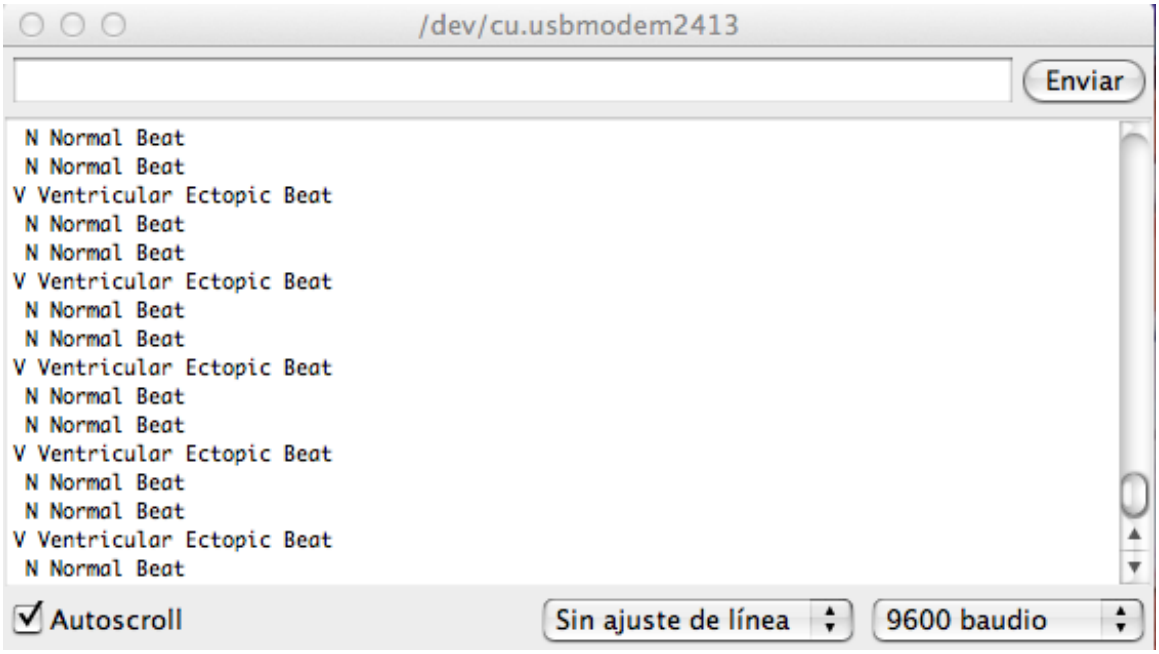


Figura 54. Resultados mostrados por pantalla al ejecutar la Red Neuronal en la plataforma Intel Edison para el registro 223 de MIT-BIH [17]

- ***Cálculo del ritmo cardíaco mediante el algoritmo Pan-Tompkins:*** Ya tenemos en la memoria de la plataforma Intel Edison guardados los registros de 10 s que serán la entrada a nuestro algoritmo. Utilizamos las funciones de código C en el entorno Arduino [9] para leer el archivo de memoria y guardarlo en un array. Hay que utilizar las funciones de Matlab Coder [43] para, en este caso, crear dos arrays dinámicos de tamaño en exceso. Esto es debido a que estos dos arrays serán las salidas de la función algoritmo de Pan-Tompkins y su tamaño, a priori, es desconocido. Si una señal tiene 10 latidos en 10 s entonces sus arrays de salida serán de 10 posiciones, pero si tiene 15 latidos los arrays de salida serán de 15, esto se soluciono creando dos arrays de tamaño 1000 que nunca serán sobrepasados porque es clínicamente imposible que un humano tenga 1000 latidos por cada 10 s, y entonces su array de salida tendría un largo de 1000. El siguiente paso es ejecutar la función algoritmo de Pan-Tompkins que dará como resultado estos dos arrays. Uno de estos arrays, que recordemos tienen ambos el mismo tamaño (uno indican índices de QRS y otro la amplitud), actúa como entrada a la función HeartRatePanT (creada en Matlab y explicada en el capítulo anterior) que calcula el ritmo cardíaco en pulsaciones por minuto y lo muestra por pantalla como vemos en la Figura 55:

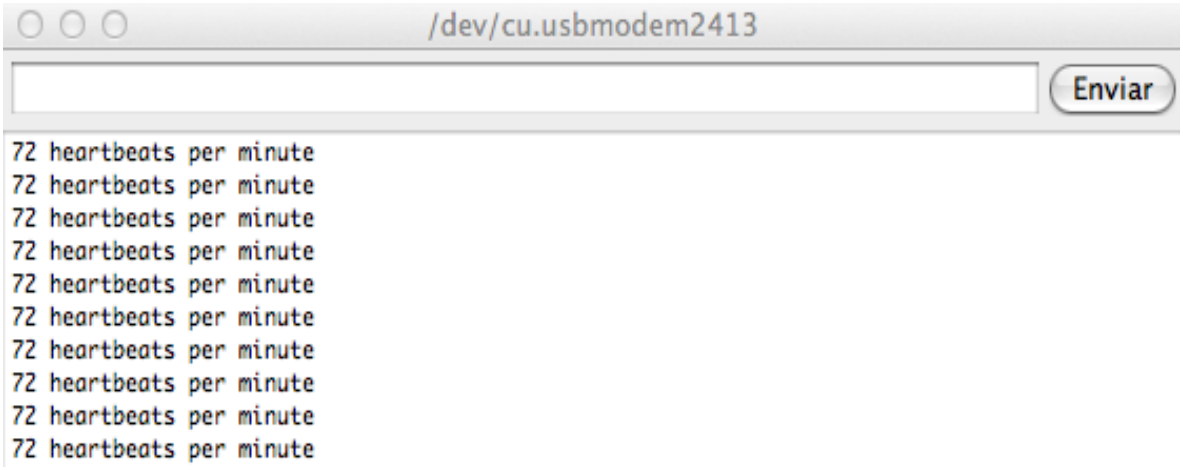


Figura 55. Resultados mostrados por pantalla al calcular el ritmo cardíaco mediante el algoritmo de Pan-Tompkins en la plataforma Intel Edison para el registro 111 de MIT-BIH [17]

7.2. Comprobación y validación de los resultados en la plataforma Intel-Edison

Es necesario comprobar que el proceso de transformación y adaptación a código C realizado para los programas diseñados en Matlab, así como la inclusión en sus archivos de nuevas funciones como las que permiten leer los archivos de la memoria de la Intel Edison, no ha modificado el comportamiento de las funciones utilizadas. Como se explico en el capítulo anterior, las funciones fueron comprobadas y validadas en Matlab pero es necesario volver a realizar este proceso al haber sido ahora implementadas en la plataforma Intel Edison y haber sido modificadas en el IDE de Arduino.

Para comprobar que los resultados mostrados por pantalla son correctos utilizaremos la página web de *Physionet* [3], que dispone de la herramienta *PhysioBank ATM ,* que permite acceder a todas las bases de datos de *Physionet* [3] para visualizar formas de onda, intervalos RR, etc, así como exportar los registros en diferentes formatos. Es una herramienta básica y una ayuda de desarrollo para todo aquel que este realizando algún proyecto con una base de datos de *Physionet* [3]*,* como en este trabajo donde los recursos de esta página han servido de gran apoyo. A continuación explicaremos como se ha comprobado el correcto funcionamiento para cada una de nuestras dos aplicaciones, el clasificador de latidos y el cálculo del ritmo cardíaco:

- ***Clasificador de latidos con Red Neuronal:*** para comprobar que los resultados mostrados por pantalla son correctos, hemos utilizado la opción de *PhysioBank ATM "show annotations as text"* que nos muestra todas y cada una de las anotaciones realizadas por los cardiólogos en cada uno de los registros de la base de datos MIT-BIH Arritmia [17]. En estas anotaciones se pueden encontrar etiquetas de ritmo, el tipo de cada latido, la calidad de las señales, etc. Escogemos el ejemplo mostrado en la Figura 54, el registro 223, y comprobamos que los latidos mostrados por pantalla coinciden con ese tramo del registro 223 como vemos en la Figura 56, donde se han combinado dos imágenes: la anterior imagen la Figura 54 y con fondo amarillo una captura de *PhysioBank ATM* donde se muestran los latidos del registro 223, así como su nº de muestra y tiempo. (La anotación + se refiere a un comienzo de ritmo normal en el ECG, (N)

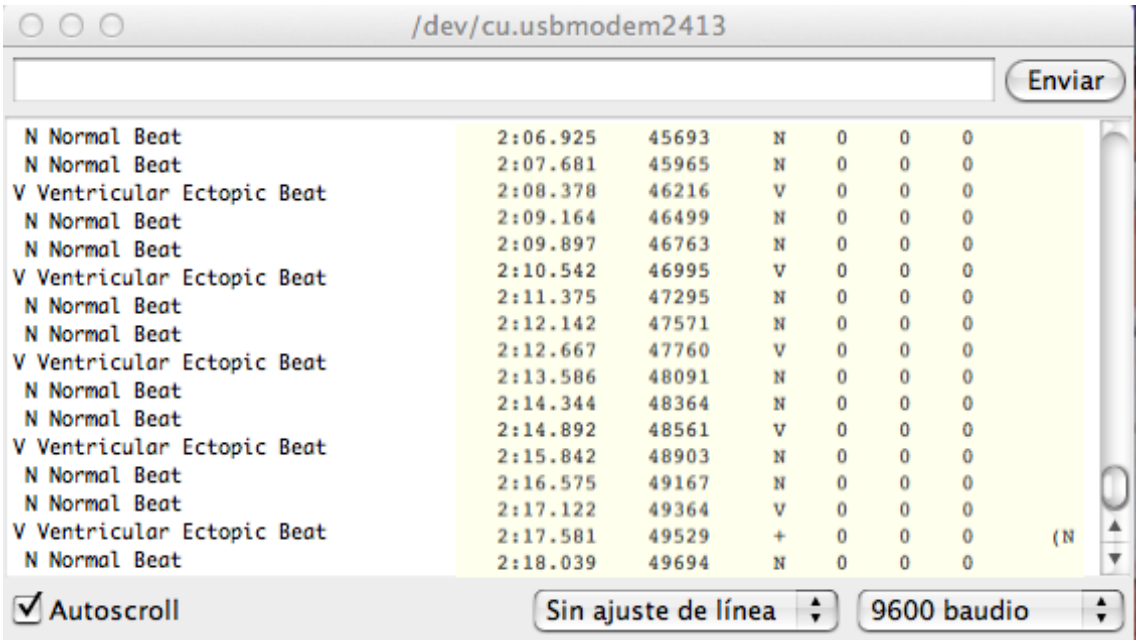


Figura 56. Comprobación de los resultados de la plataforma Intel Edison con la aplicación online *PhysioBank ATM* [3]

➢ ***Cálculo del ritmo cardíaco mediante el algoritmo Pan-Tompkins:*** para comprobar que el algoritmo Pan-Tompkins funciona correctamente implementado sobre la plataforma Intel Edison, hemos utilizado la misma opción de *PhysioBank ATM* que en el párrafo anterior para comprobar el funcionamiento del clasificador de latidos con Red Neuronal, que nos permiten comprobar cuantos latidos hay en los primeros 10 s. Tomando el ejemplo mostrado en la Figura 55, el registro 111, recordemos que su ritmo cardíaco calculado era de 72 pulsaciones por minuto. Eso quiere decir que en 10 s tendremos 12 latidos. Efectivamente, como se muestra en la Figura 57 en los 10 primeros segundos de la señal 111 tenemos 12 latidos (la primera etiqueta + es una etiqueta de ritmo, no es una etiqueta de latido), por lo tanto el resultado mostrado en pantalla por la plataforma Intel Edison es correcto.

Time	Sample #	Type	Sub	Chan	Num	Aux
0:00.086	31	+	0	0	0	(N
0:00.547	197	L	0	0	0	
0:01.358	489	L	0	0	0	
0:02.233	804	L	0	0	0	
0:03.131	1127	L	0	0	0	
0:03.972	1430	L	0	0	0	
0:04.831	1739	L	0	0	0	
0:05.667	2040	L	0	0	0	
0:06.511	2344	L	0	0	0	
0:07.336	2641	L	0	0	0	
0:08.225	2961	L	0	0	0	
0:09.106	3278	L	0	0	0	
0:09.942	3579	L	0	0	0	

Figura 57. Latidos en los primeros 10 s del registro 111 de MIT-BIH Arritmia [17]

Finalmente podemos concluir que los resultados de implementación de las funciones creadas en Matlab para la creación de dos aplicaciones (clasificador de latidos y cálculo del ritmo cardíaco) han sido satisfactorios.

Capítulo 8

Conclusiones

En este capítulo se describirán las conclusiones finales del proyecto después de evaluar el resultado final, y las posibles líneas de trabajo futuras

8.1. Resultados

Una vez se han analizado los resultados de desarrollar una aplicación para la clasificación de latidos y una calculadora de ritmo cardíaco basada en el algoritmo de Pan-Tompkins en Matlab, e implementadas sobre la plataforma Intel Edison se puede concluir que los resultados son satisfactorios puesto que ambas aplicaciones funcionan correctamente. La Red Neuronal es capaz de clasificar latido en tipos estándar con un porcentaje de error aceptable y el algoritmo de Pan-Tompkins es adecuado para la aplicación que calcula el ritmo cardíaco. Ambas son capaces de ejecutarse en la plataforma y mostrar los resultado en un PC.

8.2. Objetivos

Respecto a los objetivos del proyecto, que se han comentado anteriormente evaluando los resultados, se han cumplido con creces puesto que se ha logrado crear las dos aplicaciones diseñadas y se ha logrado que funcionen correctamente en una plataforma Intel-Edison

Respecto a los objetivos del sistema diseñado también han sido cumplido puesto que se ha comprobado y demostrado que la plataforma Intel Edison tiene memoria y capacidad de cálculo, así como el tamaño reducido necesario, para poder desarrollar en ella un sistema completo biomédico de análisis de señales ECG que permita ser llevado encima por el paciente y sin necesidad de cables. La plataforma ha demostrado su memoria siendo cargado con los archivos de la Base de Datos MIT-BIH Arritmia, ha podido procesar los datos en la Red Neuronal y permite la conexión inalámbrica mediante WiFi.

8.3. Generales

Respecto a los objetivos generales de la asignatura, Trabajo de Fin de Grado, se han cumplido todas las expectativas pues el alumno ha sido capaz de adquirir las competencias y conocimientos necesarios para diseñar y desarrollar un sistema real con bases de datos reales e implementarlos en una plataforma. El trabajo ha servido para aprobar la asignatura, así como para aprender mucho sobre distintos campos como las redes neuronales artificiales, el procesamiento de señales en la herramienta Matlab o la creación de la creación de documentos descriptivos de proyectos como es la presente memoria.

8.4. Líneas de trabajo futuras

Como se explico en los objetivos del proyecto, uno de los puntos más interesantes del trabajo es la posibilidad de aprovecharlo para ser la base en desarrollos futuros.

En primer lugar el presente proyecto puede desarrollarse en el futuro para crear una aplicación biomédica con las siguientes características: recoger datos de los electrodos de un ECG, digitalizar la señal, filtrarla, procesarla para detectar los latidos presentes, separar y extraer las características de un latido necesarias para clasificarlo, calcular el ritmo cardíaco y en definitiva ser una ayuda en el diagnóstico médico de enfermedades cardíacas. Así mismo el sistema debería ser inalámbrico y portable para poder ir siempre instalado con el paciente, y capaz de enviar datos a terminales externos sin necesidad de cables. Esto ha sido demostrado al implementarlo sobre la plataforma Intel Edison capaz de cumplir con todos los requisitos que debería tener ese dispositivo portátil.

Por otro lado algunos de los datos obtenidos en el procesamiento de señales ECG pueden servir de ayuda para otros proyectos. En concreto la Tabla 6. Valores medios de cada característica para cada tipo de latido ANSI / AAMI, puede ser utilizada como información de entrada para otro tipo de aplicaciones de clasificación de latidos, como pueden ser los clasificadores bayesianos.

Capítulo 9

Presupuesto

En este capítulo se describen las etapas en las que ha sido realizado el proyecto, así como su coste

9.1. Etapas del proyecto

El proyecto fue realizado desde el 1 de Mayo hasta el 26 de Septiembre de 2015. Se dedicaron un total de 6 días a la semana durante 4 semanas al mes, dando un total de 120 días. Con una media de 5 horas al día dedicadas al trabajo, tenemos un total de **600** horas

- **1) Planificación**

  - Estudio de diferentes opciones y aplicaciones (15 días)
  - Estudio de las redes neuronales artificiales (20 días)
  - Estudio del uso de la plataforma Intel Edison (10 días)

- **2) Desarrollo**

  - Procesamiento de señales en Matlab (30 días)
  - Diseño de la Red Neuronal (5 días)
  - Configuración de la plataforma Intel Edison (5 días)
  - Implementación en la plataforma Intel Edison (10 días)

- **3) Documentación**

  - Redacción de la memoria (25 días)

9.2. Presupuesto

Este es el coste de los recursos utilizados para el proyecto

- **Recursos software**

  - Licencia Matlab: ***6000 €***

  - Licencia Microsoft Office 365 Hogar: ***100 €***

- **Recursos hardware**

  - iMac 3,06 GHz Intel Core 2 Duo: ***1000 €***

  - Intel Edison kit for Arduino: ***163 €***

  - Conectores mini USB: ***3 €***

- **Costes de personal**

  - Ingeniero: ***30 €/hora***

- **Costes totales**

CONCEPTO	COSTE
Costes de software	**7.100,00 €**
Costes de hardware	**1.166,00 €**
Costes de personal	**18.000,00 €**
IVA (21%)	**5.515,86 €**
TOTAL	**31.781,86 €**

GLOSARIO

GUI: *Graphical User Interface*

SoC: *System On a Chip*

MCU: *Microcontroller*

CPU: *Central Processing Unit*

IDE: *Integrated Development Environment*

PE: *Process Element*

RNA: Red Neuronal Artificial

ART: *Adaptive Resonance Theory*

ANN: *Artificial Neural Network*

MIT: *Massachusetts Institute of Technology*

BIH: *Beth Israel Hospital*

ECG / EKG: Electrocardiograma

IDE: *Integrated Development Environment*

Nodo SA: Nodo Sinoauricular

MLII: Derivación Periférica Modificada II

AAMI: *Association for the Advancement of Medical Instrumentation*

ANSI: American National Standards Institute

IEEE: Institute of Electrical and Electronics Engineers

WFDB: WaveForm DataBase

Bibliografía

[1] [Online]. http://es.mathworks.com/

[2] [Online]. https://es.wikipedia.org

[3] [Online]. http://www.physionet.org/

[4] Goldberger AL,Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals.," *Circulation 101(23):e215-e220* , 2000 (June 13). [http://circ.ahajournals.org/cgi/content/full/101/23/e215]. PMID: 10851218; doi: 10.1161/01.CIR.101.23.e215.

[5] I, Moody, G. Silva, "An Open-source Toolbox for Analysing and Processing PhysioNet Databases in MATLAB and Octave" *Journal of Open Research Software 2(1):e27 [http://dx.doi.org/10.5334/jors.bi] ; 2014 (September 24).*

[6] [Online]. http://www.intel.es/content/www/es/es/do-it-yourself/edison.html

[7] Intel® Edison Compute Module Hardware Guide. (January 2015)

[8] Intel Edison, "Intel® Edison Kit for Arduino* , Hardware Guide ," February 2015.

[9] [Online]. www.arduino.cc/en/Main/Software

[10] Basogain Olabe, Xabier. Redes neuronales artificiales y sus aplicaciones. Escuela Superior de Ingenieria de Bilbao, UPV-EHU. [Online]. http://cvb.ehu.es/open_course_ware/castellano/tecnicas/redes_neuro/Course_listing.html

[11] Damián Jorge Matich, "Redes Neuronales: Conceptos Básicos y aplicaciones," Universidad Tecnologica Nacional de Rosario, Rosario, Argentina, 2001.

[12] Inés M.Galván León, Pedro Isasi Viuela, *Redes de Neuronas Artificiales, Un enfoque practico*. Madrid: Pearson, 2004.

[13] Ines M.Galvan, Jose Mª Valls, Redes de neuronas Artificiales. (2011) Open Course Ware Universidad Carlos III de Madrid. [Online]. http://ocw.uc3m.es/ingenieria-informatica/redes-de-neuronas-artificiales

[14] http://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x38.html.

[15] Dr. Dale Dubin, *Electrocardiografía práctica: lesión, trazado e interpretacion*.: McGraw-Hill.

[16] http://www.electrocardiografia.es/.

[17] Moody, Mark RG. GB, *The impact of MIT-BIH Arrhythmia Database*: IEEE Eng in Med and Biol 20(3):45-50(May-June 2001). (PMID: 11446209).

[18] Willis J. Tompkins, *Biomedical Digital Signal Processing*.: Prentice Hall, 2000.

[19] ANSI/AAMI EC57/Ed.3, *Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms*.

[20] K.M. Kempner, and M.H. Miller W.P. Holsinger, "A QRS pre-processor based on digital differentiation" *IEEE Transactions on Biomedical Engineering* , 1971.

[21] J. Pan and W.J. Tompkins., "A Real-Time QRS Detection Algorithm" *IEEE Transactions on Biomedical Engineering. Vol. 32, Issue 3, pp. 230-236. 1985.*.

[22] P.S. Hamilton and W. J.Tompkins, "Quantitative investigation of QRS detection rules using the MIT/BIH arrhythmia database ," *IEEE Transactions on Biomedical Engineering. Vol. 33, Issue 12, pp. 1157-1165. 1986.*.

[23] M. O'Dwyer, and R. Reilly, P. de Chazal, . *Automatic classification of heartbeats using ECG morphology and heartbeat interval features*: IEEE Trans. Biomed. Eng., vol. 51, no. 7, pp. 1196–1206, Jul. 2004..

[24] Richard B. Reilly, Philip de Chazal, "A Patient-Adapting Heartbeat Classifier Using ECG Morphology and Heartbeat Interval Features " *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, VOL. 53, NO. 12, DECEMBER 2006*.

[25] B. V. K. Vijaya Kumar, Miguel Tavares Coimbra, Can Ye, "Heartbeat Classification Using Morphological and Dynamic Features of ECG Signals" *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, VOL. 59, NO. 10, OCTOBER 2012*.

[26] ANSI ECAR:1987, "Recommended practice fot testing and reporting performance results of ventricular detection algorithms," AAMI, Arlington, VA, 1987,

[27] Frédéric Leclerc, Cedric Dumez-Viou, and Guy Lamarque Philippe Ravier, "Redefining Performance Evaluation Tools for Real-Time QRS Complex Classification Systems" *IEEE TRANSACTIONS ON BIOMEDICAL*

ENGINEERING, VOL. 54, NO. 9, SEPTEMBER 2007.

[28] F. Johnson, M. Rachniowski, T.H. Yeap, "ECG Beat Classification by a Neural Network ," *AnnualIntemational Conference of the IEEE Engineeringin Medicine and Biology Society,Vol. 12,No.3,1990*.

[29] http://almostadoctor.co.uk/.

[30] H. A. Moghadam M. Mohebbi, "Real-Time Ischemic Beat Classification Using Backpropagation Neural Network" K.N.Toosi University of Technology, Electrical Engineering Department ,.

[31] http://ekg.academy/.

[32] Martin T. Hagan, Howard B. Demuth Mark Hudson Beale, *Neural Network Toolbox™ User's Guide*.: Matlab, 2015b.

[33] http://es.mathworks.com/help/nnet/ug/choose-a-multilayer-neural-network-training-function.html.

[34] Philip de Chazal, MARCS Institute, University of Western Sydney, Sydney, Australia. , "A switching Feature Extraction System for ECG Heartbeat Classification" *Computing in Cardiology 2013; 40:955-958*.

[35] C. Peterson, G. Braccini, L. Edenbrandt, and L. Sornmo, M. Lagerholm, "Clustering ECG complexes using Hermite functions and self-organizing maps" *IEEE Trans. Biomed. Eng., vol. 47, no. 7, pp. 838–848, Jul. 2000.*

[36] L. T. Hoa, and T. Markiewic, S. Osowski, "Support vector machine-based expert system for reliable heartbeat recognition" *IEEE Trans. Biomed. Eng., vol. 51, no. 4, pp. 582–589, Apr. 2004.*

[37] A. Goni, and A. Illarramendi J. Rodriguez, "Real-time classification of ECGs on a PDA" *IEEE Trans. Info. Tech. Biomed., vol. 9, no. 1, pp. 23–34, Mar. 2005.*

[38] Complete Pan Tompkins Implementation ECG QRS detector. (march 2014) http://www.mathworks.com/.

[39] *Code Generation from MATLAB User's Guide*: MathWorks, vol. R2011b.

[40] Convert PAN-TOMPKINS Algorithm to C Using MATLAB Coder. [Online]. http://www.mathworks.com/

[41] "Guide, Intel® Edison Wi-Fi," February 2015.

[42] [Online]. http://es.mathworks.com/help/matlab/ref/dlmwrite.html

[43] [Online]. http://es.mathworks.com/help/fixedpoint/ug/c-code-interface-for-unbounded-arrays-and-structure-fields.html