
UOC
Universitat Oberta
de Catalunya

Adquisición, caracterización, entrenamiento y desarrollo de un clasificador mediante una señal EMG

Francisco Javier Ripoll Esteve

Máster Bioestadística y Bioinformática

Bioinformática estructural y diseño de aplicaciones bioinformáticas

Tutores:

- **Dra. Elisabeth Ortega Carrasco – UOC**
- **Dr. José Alberola Rubio – IIS La Fe**

Profesor responsable de la asignatura: D. David Merino Arranz

Enero 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Adquisición, caracterización, entrenamiento y desarrollo de un clasificador mediante una señal EMG*
Nombre del autor:	*Francisco Javier Ripoll Esteve*
Nombre del consultor/a:	*Dra. Elisabeth Ortega Carrasco – UOC* *Dr. José Alberola Rubio – IIS La Fe*
Nombre del PRA:	*D. David Merino Arranz*
Fecha de entrega (mm/aaaa):	01/2020
Titulación::	*Máster Bioestadística y Bioinformática*
Área del Trabajo Final:	*TFM-Bioinformática i Bioestadística*
Idioma del trabajo:	*Castellano*
Palabras clave	*EMG, Machine Learning, Biosignals*

Resumen del Trabajo

Este trabajo tiene por finalidad la construcción de un clasificador basado en patrones de movimiento embebido en un sistema Arduino a través de señales de electromiografía. El trabajo está dividido en tres fases. La primera de ellas es la adquisición de las señales generadas por el Shield EKG-EMG construyendo un conversor analógico digital. Además, se crea una aplicación en C# para poder almacenar los datos en una base de datos relacional. La segunda fase consiste en la preparación de los datos para la extracción de características de la señal basadas en el análisis temporal de la señal. Con este vector de características se entrenará un modelo basado en el análisis Discriminante Lineal de Fisher y evaluando todas las posibles combinaciones de características para obtener la que nos dé una mayor puntuación. La última fase concluirá implementando los mejores modelos en una placa Arduino UNO utilizando la estrategia del mayor voto. El resultado obtenido es un prototipo de clasificación en tiempo real implementado en un sistema Arduino en el que nos clasificará dos tipos de posiciones del brazo y mano según la señal de electromiografía obtenida desde el bíceps.

Abstract:

The purpose of this work is to build a classifier based on movement patterns embedded in an Arduino system through electromyography signals. The work is divided into three phases. The first of the phases is the purchase of the signals generated by the EKG-EMG Shield by building a digital analog converter. Also, an application is created in C# to be able to store the data in a relational

database. The second phase consists of preparing the data for the extraction of signal characteristics based on the temporal analysis of the signal. With this feature vector, a model based on Fisher´s Linear Discriminant analysis will be trained, and all posible combinations of features evaluated to obtain the highest score. The final phase will conclude by implementing the best models on an Arduino UNO board using the Major voting strategy. The result obtained is a real-time classification implemented prototype in an Arduino system in which we will classify two types of arm and hand positions according to the electromyography signals obtained from the biceps.

Índice

Lista de figuras

1 Introducción

Desde hace uno años, los trabajos de investigación para desarrollar interfaces hombre-máquina han cobrado mucha importancia, ya sea pensado para realizar tareas sin contacto directo del operario o para ofrecer alternativas a personas con discapacidades[1][2]. Una de las formas para conseguir estos fines es con la captación de señales de electromiografía (EMG) que provienen de músculos asociados cualitativamente con el tipo de movimiento producido. En la actualidad, los trabajos de esta área están dirigidos a encontrar los algoritmos de procesamiento, caracterización y clasificación de patrones de estas señales que permitan el análisis y la determinación del movimiento que se está produciendo[3].

Este trabajo tiene las siguientes fases:

1) Implementación de un sistema de adquisición de captación de señales de electromiografía para la construcción de una base de datos que almacene dichas señales de una forma relacionada y compacta.

2) Análisis y preparación de los datos obtenidos en la fase anterior. Caracterización de los datos a través de su análisis temporal.

3) Obtención un algoritmo de clasificación basado en el método de *Análisis Discriminante Lineal (LDA).*

4) Encapsulación del algoritmo obtenido en el punto anterior en un microcontrolador Arduino Uno.

En la Ilustración 1 se puede observar el esquema general del trabajo.

Ilustración 1. Esquema del proyecto

Todos los desarrollos han sido realizados con las siguientes herramientas de software: Microsoft Visual Studio, Microsoft SQL Server, Arduino IDE, Matlab y R.

Ilustración 2. Software utilizado

1.1 Contexto y justificación del Trabajo

Una de las dificultades que podemos encontrar en la actualidad cuando se desarrolla un modelo de clasificación en universos interactivos es llegar a encapsular el modelo resultante en un dispositivo. Si nos centramos en las señales que provienen desde un origen fisiológico, también llamadas Bioseñales[4], debemos ser capaces de poder programar un dispositivo capaz de leer dichas Bioseñales en tiempo real[5]. El término “tiempo real”, aquí, hace referencia a procesar una Bioseñal, extraer sus características, utilizarlas como parámetros de entrada en un clasificador programado en el microcontrolador y así obtener en un tiempo razonable el resultado de la clasificación.

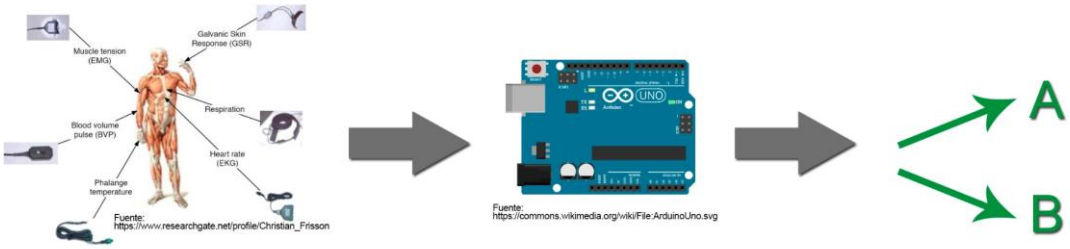


Ilustración 3. Proceso de procesamiento Bioseñal

Entre los objetivos del Máster se encuentra el de “Conocer los principales métodos estadísticos para el análisis de datos y dominar los métodos de visualización”. En concreto, algunos de los objetivos y competencias de la asignatura “Machine learning” son las siguientes:

- Identificar los diferentes tipos de problemas de clasificación que surgen a partir de datos de origen biológico.
- Conocer los principios básicos de las técnicas de clasificación.
- Comprender los métodos para el entrenamiento de clasificadores.
- Conocer las medidas de evaluación de clasificadores y saber evaluar cuando un clasificador es mejor que otro y por qué.
- Conocer los posibles problemas que pueden ocurrir en el aprendizaje de un clasificador, saber si se pueden solucionar y saber cuáles son las soluciones que se pueden aplicar.
- Ser capaces de entrenar y testear clasificadores.
- Conocer y ser capaz, a nivel general, de utilizar los principales métodos de minería de datos, así como sus aplicaciones en la biología.
- Capacidad de analizar un problema de bioinformática y ser capaz de identificar y definir los requerimientos informáticos y estadísticos apropiados para resolverlo.

El TFG se engloba dentro del área Bioinformática estructural y diseño de aplicaciones bioinformáticas de las “Áreas de TFM (Junio 2019) - Máster Bioinformática y Bioestadística UOC-UB”. La elección de esta área es interesante desde el punto de vista práctico de la asignatura “Machine learning” ya que cualquier ejercicio realizado en la asignatura finaliza con la evaluación y rendimiento de los modelos obtenidos de los datos proporcionados en el material de la propia asignatura. Por tanto, con este TFG se pretende ir más allá de lo visto en la asignatura y en el Máster. Por un lado, reemplazar los Dataset de origen por datos obtenidos y procesados por mí mismo, como realmente se realizaría en un estudio de verdad; por otro lado, ser capaz de encapsular los modelos obtenidos en la asignatura en un dispositivo.

1.2 Objetivos del Trabajo

1.2.1 Objetivos generales

- Conocer los principios básicos de los bioseñales y su adquisición.
- Ser capaz de entrenar, testear y evaluar el rendimiento de distintos modelos de clasificación.
- Conocer los métodos de encapsulación de modelos de clasificación en un microcontrolador.

1.2.2 Objetivos específicos

- Adquirir y tratar un bioseñal mediante métodos de conversión analógico-digital.
- Realizar la construcción de un Dataset estructurado y de calidad.
- Llevar a cabo la exploración y preparación de los datos según el algoritmo de clasificación deseado.
- Obtener modelos de clasificación mediante los datos de entrenamiento.
- Evaluar el rendimiento del modelo con los datos de prueba.
- Realizar la programación en un microcontrolador para la encapsulación de los modelos obtenidos.

1.3 Enfoque y método seguido

El enfoque empleado en este trabajo es el tipo de Investigación aplicada, ya que lo que queremos es convertir el conocimiento teórico de la adquisición de una señal en una herramienta de comunicación.

La técnica de recolección de datos será mediante observación en cada individuo, donde en una base de datos se registrarán las siguientes variables básicas de cada individuo: Nombre, Apellido, Sexo, Fecha de nacimiento, Peso y Altura mediante una aplicación informática desarrollada para este propio proyecto.

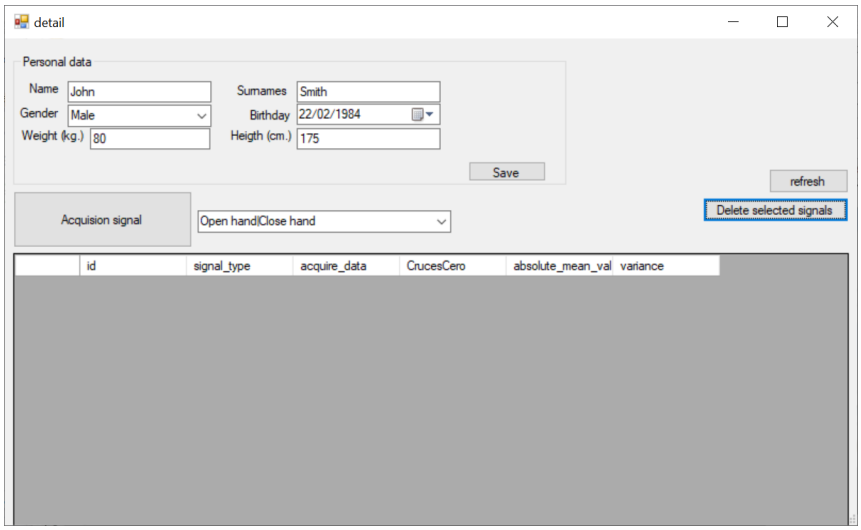


Ilustración 4. Aplicación para registrar la base de datos de individuos y muestras

Mediante una conversión de los valores físicos en voltaje recogidos por los electrodos, se realizará una conversión digital de estos datos donde se almacenarán en esta misma aplicación, registrando fecha y hora de su captura.

Por cada individuo se registrará un registro completo de cada protocolo establecido. Cada protocolo tendrá definido el número de posiciones a registrar *NP*. Por cada posición *POS*, registraremos una muestra *M* de 5 segundos, que iremos alternando hasta realizar 5 repeticiones. Para poder estabilizar el sistema, se dejará tanto al principio como al final 5 segundos de estabilización del sistema.

Cada protocolo tendrá la siguiente estructura:

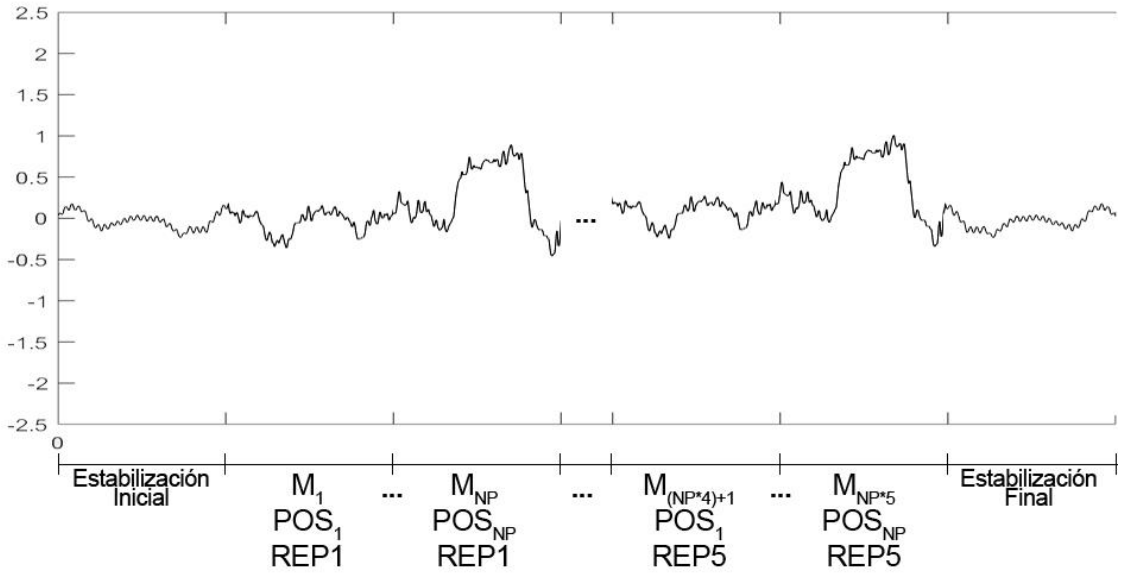


Ilustración 5. Estructura protocolo

La duración de cada protocolo será la siguiente:

$$t_{protocolo} = 5 + 5 * 5 * (NP) + 5$$

Ecuación 1

1.4 Planificación del Trabajo

Durante el desarrollo de dicho trabajo, se han realizado varias pruebas de evaluación continua (en valenciano “proves d’avaluació continua” o PAC) de entrega en la asignatura. En una de estas pruebas de evaluación iniciales se detallaron las tareas a realizar y la confección de una planificación temporal mediante un diagrama de Gantt. En las siguientes PAC, aunque las tareas no han variado en su esencia, la planificación sí que ha ido variando por diversos motivos que se han ido detallando en los distintos entregables.

El diagrama de Gantt se puede ver en el anexo 1. El detalle de las tareas a realizar en dicho TFM son las siguientes:

- Memoria TFM: Redacción de la memoria del trabajo final de máster. Esta tarea estará compuesta por la propia redacción de la memoria, que se irá realizando en paralelo con el proyecto para no tener una carga elevada al final del proyecto y la PAC4 que coincide con el cierre de la memoria.

- PAC0 – Propuesta TFM: Entrega de la prueba de evaluación continua PAC0, que consiste en la redacción del documento donde se reflejan los aspectos iniciales del trabajo acordado con el/la directora/a.

- PAC1 – Plan de trabajo: Entrega de la prueba de evaluación continua PAC1, que consiste en la redacción del documento actual, el plan de trabajo.

- Desarrollo del trabajo: Esta tarea será la más larga del proyecto, ya que durante este periodo se llevará a cabo el desarrollo del trabajo. Se compone de las siguientes subtareas:

    - Estudio del Hardware: Lectura de manuales y búsqueda de información sobre Arduino Uno, Olimex Shield EKG-EMG, adquisición de señales y conversión ADC.

    - PAC2 y PAC3 – Desarrollo del trabajo – Fase 1/2: Entrega de la prueba de evaluación continua PAC2, que consiste en la entrega de un seguimiento periódico del desarrollo del trabajo realizado hasta la fecha.

    - Adquisición de datos: En esta tarea realizaremos aquellos trabajos necesarios para la correcta adquisición de las muestras de los individuos y su correcta preparación de los datos.

        - Desarrollo aplicación adquisición datos.

        - Obtención y evaluación de muestras de prueba.
        - Obtención de las muestras de individuos.
        - Preparación de los datos.
    - Análisis de los datos: Con los datos preparados se realizará el entramiento, test y correspondientes evaluaciones de los modelos obtenidos.
    - Encapsulación de los modelos: Llegados a este punto, se necesitará tener los modelos obtenidos para su encapsulación en el microcontrolador de Arduino.
    - Test Funcionamiento Real: Se realizarán distintas pruebas para analizar que la herramienta programada en el Arduino funciona correctamente.

- PAC5a – Elaboración de la presentación: Entrega de la prueba de evaluación continua PAC5a, que consiste en la elaboración de la presentación.
- PAC5b – Defensa pública: Presentación y defensa del trabajo delante del tribunal.

1.5 Breve sumario de productos obtenidos

El producto final obtenido es el prototipo de un dispositivo clasificador basado en patrones de movimiento provenientes desde una señal de electromiografía. Este prototipo indica mediante dos Leds la clasificación obtenida al mismo tiempo que manda el resultado de la clasificación mediante el cable serial.

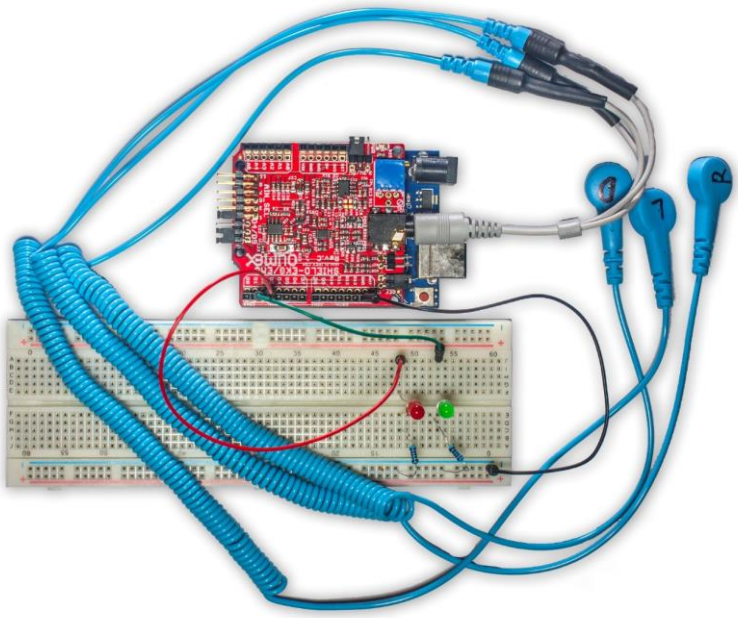


Ilustración 6. Resultado obtenido.

1.6 Repositorio de ficheros

Para que todo este trabajo pueda ser reproducible, se ha subido en un repositorio de GitHub:

- Dataset con los datos obtenidos en la fase de adquisición de la señal
- Código fuente del programa C# para el almacenamiento de los datos obtenidos desde el ADC.
- Script para R para la transformación de los datos SQL Server en ficheros CSV
- Archivos CSV resultantes
- Scripts y funciones para Matlab utilizados en la preparación, caracterización y entrenamiento del modelo.
- Sketchs para Arduino, tanto para el ADC de la primera fase como el resultado final del proyecto.
- Librería Features_EMG para C++.

Se puede acceder al repositorio a través del siguiente enlace https://github.com/ripollete/TFM_EMG.

Durante el trabajo, se hará referencia a dichos ficheros.

2 Conceptos previos

2.1 Bioseñales

La señal es una cantidad que varía en el tiempo y que contiene información. El cuerpo humano genera una cantidad de señales fisiológicas que pueden adquirirse mediante distintas técnicas y mecanismos. Estas señales pueden informarnos sobre el estado de este, ya que un defecto o una alteración genera diferentes señales.

Existen multitud de Bioseñales médicas, las cuales pueden agruparse como: biopotenciales, mecánicas, acústicas, imágenes, impedancias, señales biomagnéticas y señales bioquímicas[6].

2.2 Electromiografía

Algunas células denominadas excitables, muestran una característica de la producción de potenciales bioeléctricos como desenlace de su actividad electroquímica de las membranas, ya sean nerviosas, musculares y del tejido glandular[6].

A consecuencia que cada tipo de célula presenta una actividad eléctrica con distinta característica, el resultado de la medición nos ofrece información sobre su funcionamiento. Puesto que las disfunciones se evidencian en la señal bioeléctrica, se puede obtener información para diagnosticar a partir de esos registros.

La **electromiografía** (sus siglas EMG) se encuentra dentro de la categoría de bioseñal Biopotencial, ya que esta estudia la actividad eléctrica muscular. La activación de cada fibra del músculo se produce como respuesta a un potencial de acción transmitido por medio de la fibra nerviosa motora (axón), que inerva la fibra muscular. La composición de la célula nerviosa motora en la espina dorsal, su axón, y las fibras musculares que inerva, forma la unidad funcional básica del sistema muscular, que se denomina unidad motora. Cuando el potencial de acción nervioso obtiene la unión compuesta de tejido especializado entre el nervio y el músculo, cierta cantidad de transmisor químico se produce, cambiando el potencial de acción nervioso en otro muscular que se propaga por la fibra muscular completa.

La EMG se considera[5]:

- Señal permanente: La señal no es inducida externamente.
- Señal dinámica: La seña varía mucho en el tiempo
- Señal eléctrica.

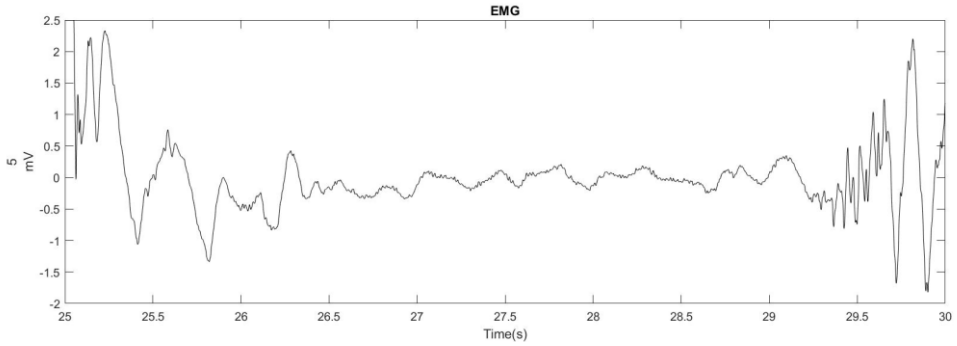


Ilustración 7. Ejemplo señal EMG

El rango de amplitud de la señal EMG es de 0.1 – 5 mV [6][7][1] y el rango de la frecuencia de 0Hz a 500 Hz [5].

Existen tres tipos de electromiografía: Electromiografía de fibra única (SFEMG: Single Fiber Electromyography); Potencial de acción de la unidad motora (MUAP: Motor unit action potential) y EMG de superficie (SEMG).

Este trabajo se centrará en el último tipo, la señal electromigráfica de superficie que se obtendrá mediante electródos superficiales situados sobre la piel. La información que se obtiene por medio de este tipo de señal es la actividad eléctrica total asociada con la contracción muscular. El ancho de banda para músculos estriados es de 2-500 Hz, y para músculos lisos 0.01-1Hz [8][9][10][11][12].

2.3 Conversión señales analógico digital

Las señales analógicas son aquellas señales generadas por algún tipo de fenómeno electromagnético y puede ser representada por una función matemática continua. Existen multitud de sistemas analógicos como, por ejemplo: un sensor de luz (LDR) o un micrófono.

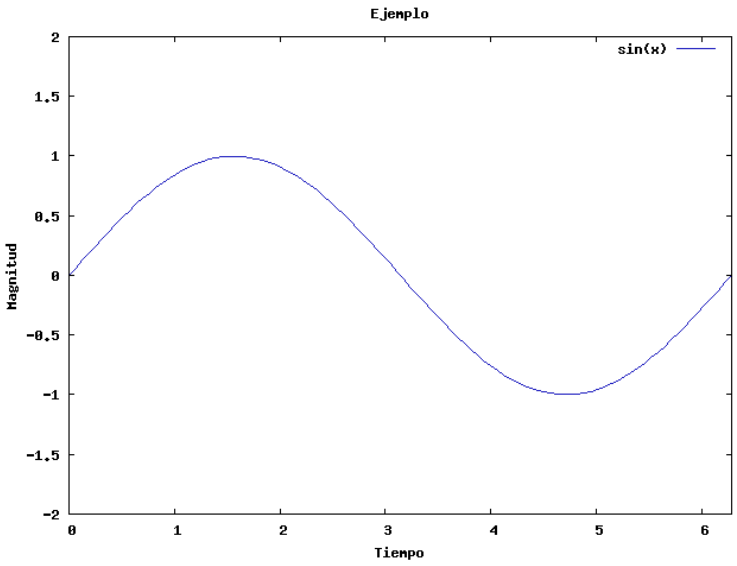


Ilustración 8. Señal analógica continua.

El mundo real es análogo[13], y por tanto se necesita poder enlazar estas variables analógicas en sistemas y procesos digitales. Estas conversiones pueden ser:

- ADC: Conversión de una señal analógica a digital.
- DAC: Conversión de una señal digital a analógica.

El proceso de convertir una señal analógica no es más que transformar una señal analógica en un número digital equivalente, y la conversión digital a analógico es transformar un número digital en una señal analógica.

Para poder realizar esta conversión y sea lo más fiel posible y no perder información, se deben de tener en cuenta los siguientes parámetros: frecuencia de muestreo, resolución y rangos de entrada…

Centrándonos en la ADC, los transductores permiten relacionar señales del universo en señales eléctricas analógicas. En la Ilustración 9, se puede ver la secuencia de un sistema físico del universo desde que entra en el sistema hasta que se transforma en código binario.

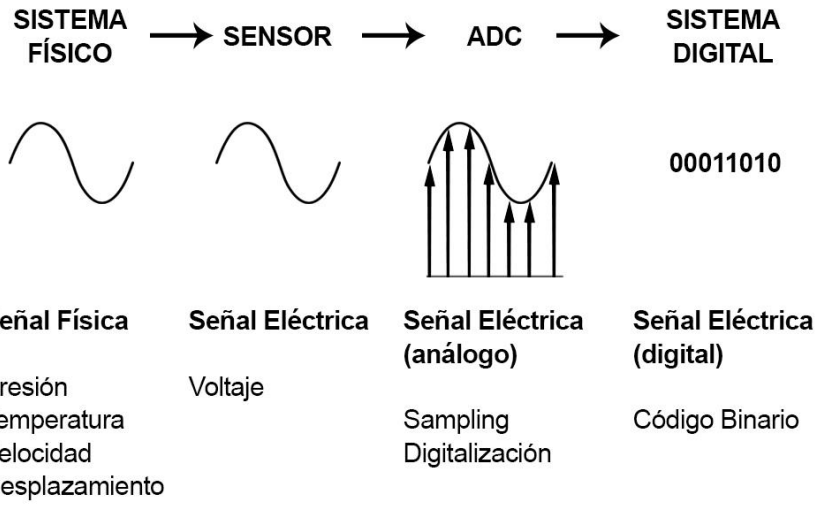


Ilustración 9. Conversión analógico digital

Para que esta señal pueda ser ingresada en el sistema digital, se deben de tomar valores discretos en momentos distintos del tiempo de la señal analógica, lo que se llama el *muestreo*.

2.3.1 Muestreo

El muestreo digital consiste en tomar muestras del valor de la señal n veces por segundo, obteniendo como resultado n niveles de tensión en un segundo. El muestreo está basado en el *Teorema de muestreo de Nyquist-Shannon*, que no debe confundirse en el siguiente paso que es la cuantificación que explicaremos más adelante.

Cuando se adquiere una señal analógica que es lo que ocupa en este caso, se tiene como objetivo almacenar la mínima información necesaria para que posteriormente un aparato electrónico pueda reconstruir la señal original para poder interpretar la información.

Si a una señal sinodal como la que se puede observar en la Ilustración 10 le vamos eliminando tramos, visualmente podemos continuar reconociendo la imagen original. Pero ¿cuántos tramos podemos eliminar de información para ser capaces de poder volver a reconstruir la señal sinodal original?

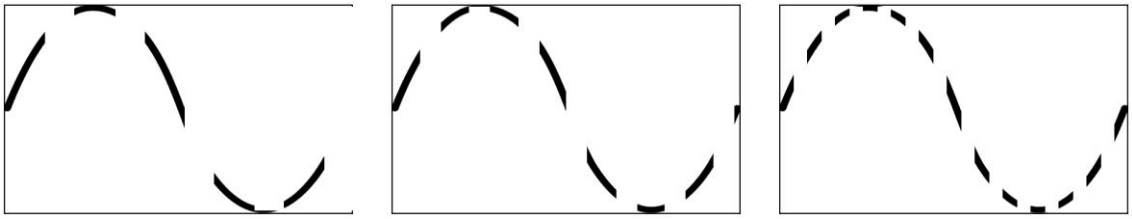


Ilustración 10. Eliminación de información onda sinodal

En 1924, Nyquist demostró que no es necesario enviar una sucesión infinita de puntos para reproducir la señal analógica para que una vez que esté digitalizada pueda reproducirse y ser interpretada, sino que basta con solo dos muestras por ciclo para que se pueda reproducir e interpretar la señal original[14].

Dicho formalmente: “Toda señal limitada en banda (debido a que se encuentra acotada por el AB del canal) se puede reconstruir completamente a partir de las muestras tomadas de misma, siempre que la velocidad del muestreo se realice como Señal Original Valores Muestreados mínimo al doble de la máxima frecuencia de la señal. A esta velocidad de muestreo se la denomina *Frecuencia de Nyquist*” (Semeria, 2015, págs. 10-11).

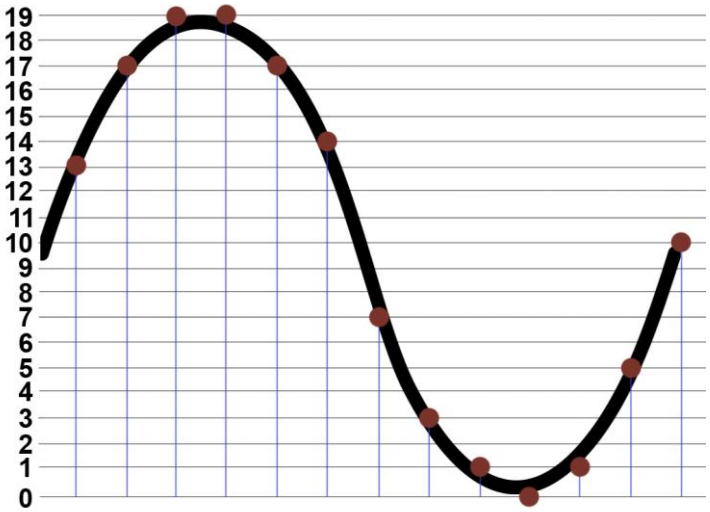


Ilustración 11. Ejemplo muestreo

2.3.2 Cuantificación

La cuantificación es la etapa posterior al muestreo en la que se han tomado los valores de la señal analógica. Esta etapa consiste en asignar un determinado valor discreto a cada uno de los niveles obtenidos en el muestreo.

Como en el universo los niveles de las señales analógicas puedes ser infinitos, lo que se realiza es una aproximación al valor más próximo en una serie de valores predeterminados, para que así la señal digital pueda ser representada por un valor finito para posteriormente se transforme en una sucesión de unos y ceros.

2.4 Algoritmos de clasificación

Desde que nacemos, los sensores de nuestro cuerpo son continuamente inundados por datos en bruto y nuestro cerebro traduce en imágenes, sonidos, olores, sabores y texturas [15]. Ya sea mediante el lenguaje verbal o no verbal, podemos compartir estas experiencias con otras personas.

Si nos remontamos atrás en la historia, los primeros astrónomos registraban la alineación de los planetas y estrellas, o las ciudades registraban los pagos de impuestos, nacimientos y muertes… Actualmente, estas observaciones y muchas más están automatizadas y se registran sistemáticamente en bases de datos computarizadas.

Como se ha comentado, con la llegada de los sensores electrónicos que son capaces de ver, escuchar, oler, probar y sentir procesan los datos de una forma muy distinta a la de los humanos, ya que estos equipos electrónicos nunca toman un descanso y nunca dejan que su juicio sesgue su percepción.

Lantz (2013, pág. 2) nos indica que, aunque el diluvio de datos a llevado a afirmar que se ha entrado en una era del Big Data, los seres humanos siempre han estado rodeados de grandes cantidades de datos [15].

El campo que estudia el desarrollo de algoritmos informáticos para transformar los datos de entrada en acciones inteligentes se conoce como *aprendizaje automático*.

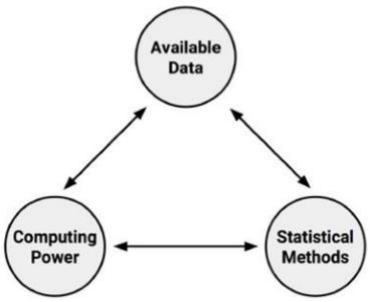


Ilustración 12. Aprendizaje automático

Los algoritmos de clasificación son tareas llevadas a cabo por los Sistemas Inteligentes ya sea por paradigmas desarrollados por la Estadística (Regresión

Logística, Análisis Discriminante) o bien por la Inteligencia Artificial (Redes Neuronales, Inducción de Reglas, Árboles de Decisión, Redes Bayesianas) y son capaces de realizar tareas de clasificación [16].

2.5 Algoritmo DLA

El análisis Discriminante Lineal (LDA por sus siglas en inglés) es una generalización de un método utilizado en reconocimiento de patrones, estadística y aprendizaje de máquinas para encontrar una combinación lineal de rasgos que singulariza o aleja dos o más clases de objetos o eventos. Se trata de la generalización de la discriminante lineal de Fisher [17].

Esta técnica se utiliza como un clasificador lineal o para la reducción de dimensiones antes de la posterior clasificación. LDA separa las clases al dibujar una región entre las diferentes clases para su decisión e intenta maximizar la relación de la varianza entre clases y la varianza dentro de clase [18].

2.6 Características de interés (señal)

Existen dos técnicas distintas de análisis para el procesamiento y extracción de señales según sea su análisis temporal o espectral [19]. Para el cálculo de las características basadas con el análisis temporal el coste computacional no es muy costoso, ya que no implica la transformación de la señal. El análisis espectral implica la transformación de la señal de tiempo al dominio de la frecuencia, utilizando técnicas como la transformada rápida de Fourier (FFT) o el Periodograma, técnicas que implican un coste computacional elevado.

3 Diseño del protocolo

El enfoque empleado en este trabajo es el tipo de Investigación aplicada, ya que lo que se ha realizado es convertir el conocimiento teórico de la adquisición de una señal en una herramienta de comunicación.

La actividad muscular es una realidad observable, medible y que se puede percibir de manera precisa, por lo tanto, la metodología utilizada será cuantitativa, siendo una de las dos metodologías de investigación que tradicionalmente se han utilizado en las ciencias empíricas. Esta metodología se centra en los aspectos observables susceptibles de cuantificación, y utiliza la estadística para el análisis de los datos.

Esta herramienta de comunicación se debe de entrenar con datos, y para ello se han definido las siguientes posiciones del brazo y de la mano que serán las que se utilicen para entrenar el modelo y posteriormente el dispositivo debe ser capaz de clasificar según unos parámetros de entrada en que posición nos encontramos.

Las distintas posiciones que se van a utilizar son:

- BEMA: Brazo extendido con la mano abierta.
- BCMC: Brazo contraído con la mano cerrada.
- BEMC: Brazo extendido con la mano cerrada.

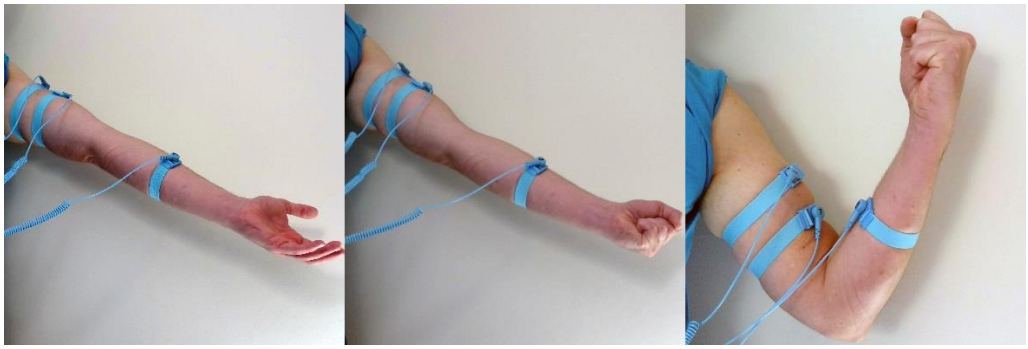


Ilustración 13. Posiciones del brazo y la mano utilizadas en el trabajo

Se han definido dos protocolos para la adquisición de las muestras que se han tomado los individuos, el protocolo A y el protocolo B. El protocolo A, únicamente registrará las posiciones *Brazo Extendido con la mano abierta* (BEMA) y *Brazo Contraído con la mano cerrada* (BCMC). En el protocolo B, además de las dos posiciones del protocolo A, añadirá la posición *Brazo extendido con la mano cerrada* (BEMC).

Cada una de estas posiciones se repetirán 5 veces de forma alterna a lo largo del registro de su correspondiente protocolo, en el que, además, añadiremos 5 segundos tanto al inicio como al final del registro donde el individuo mantendrá una posición BEMA para estabilización del sistema.

La duración de cada uno de los registros por cada protocolo es el siguiente:

- Protocolo A: 5 segundos estabilización + (2 posiciones * 5 repeticiones * 5 segundos) + 5 segundos de estabilización = 60 segundos.
- Protocolo B: 5 segundos estabilización + (3 posiciones * 5 repeticiones * 5 segundos) + 5 segundos de estabilización = 85 segundos.

Las posiciones tanto del brazo como de la mano para cada uno de los protocolos es la siguiente:

TABLA 1 PROTOCOLOS

PERIODO DE TIEMPO	PROTOCOLO A	PROTOCOLO B
0-5	Estabilización del sistema	
5-10	BEMA	BEMA
10-15	BCMC	BEMC
15-20	BEMA	BCMC
20-25	BCMC	BEMA
25-30	BEMA	BEMC
30-35	BCMC	BCMC
35-40	BEMA	BEMA
40-45	BCMC	BEMC
45-50	BEMA	BCMC
50-55	BCMC	BEMA
55-60	Estabilización del sistema	BEMC
60-65		BCMC
65-70		BEMA
70-75		BEMC
75-80		BCMC
80-85		Estabilización del sistema

Para poder tener una mayor organización de los datos, se ha realizado una aplicación en C# .net que se conecta a una base de datos SQL Server. El código fuente y el archivo de base de datos SQL Server está disponible en el siguiente enlace https://github.com/ripollete/TFM_EMG/tree/master/signalAcquirer. La aplicación asigna un número de individuo único que será el que se utiliza para anonimizar de forma disociada reversible [20] los datos que se utilizan en todo el proceso. Esta aplicación sirve para conectarse al microcontrolador y mediante

una conversión de los valores físicos en voltaje recogidos por los electrodos, realizará una conversión digital de estos almacenándolos en la misma aplicación, donde registramos fecha, hora, frecuencia de muestreo y tipo de protocolo de cada captura.

La técnica de recolección de los datos ha sido mediante observación en cada individuo, donde en una base de datos a través de la aplicación informática se han registrado las siguientes variables básicas de cada individuo: Nombre, Apellido, Sexo, Fecha de nacimiento, Peso y Altura como se puede apreciar en la Ilustración 4.

Los instrumentos que se han utilizado para todo este proceso de adquisición de la señal han sido:

- Arduino UNO.
- Shield EKG-EMG de Olimex.
- Shield EKG-EMG_PA de Olimex.
- Ordenador

Los pasos e implicaciones que han tenido cada uno de estos instrumentos se explicaran más adelante en el apartado del desarrollo del trabajo.

3.1 Población y muestra

La población inicial estimada era de 100 individuos, pero debido al tiempo que ha implicado cada adquisición, la población final ha sido de 51 individuos. De los cuales 27 individuos son de sexo masculino y 26 individuos de sexo femenino.

La distribución de la población en cuanto a las edades es la siguiente:

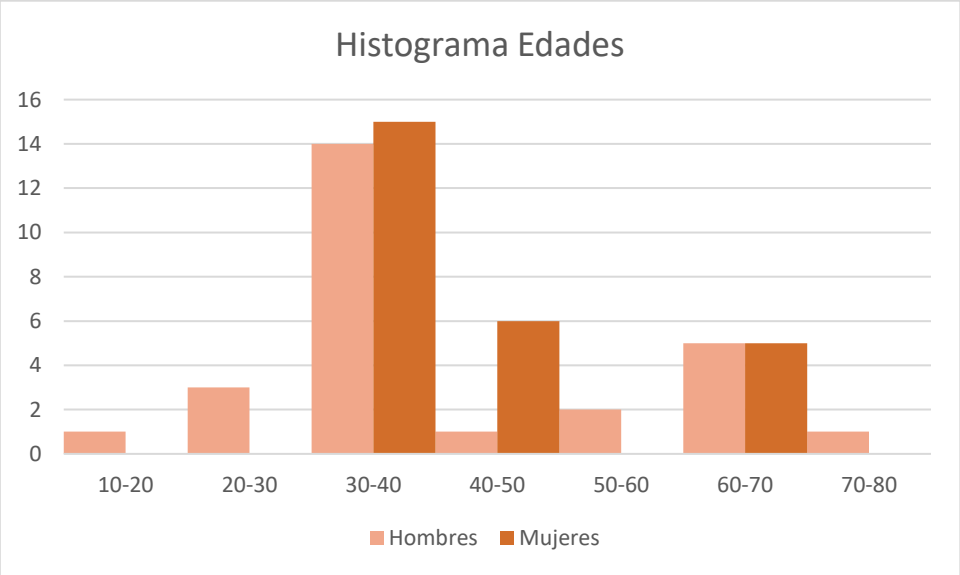


Ilustración 14. Histograma de edades

Como se han recogido los datos del peso y la estatura de los individuos, con la siguiente formula, se puede calcular el índice de masa corporal:

$$IMC = \frac{peso[kg]}{estatura(metros)^2}$$

Según la página web de la Sociedad Española para el Estudio de la Obesidad, la clasificación según el IMC es la siguiente:

Tabla 2. Clasificación IMC

IMC	CLASIFICACIÓN
<18,5	Peso insuficiente
18,5-24,9	Normopeso
25-26,9	Sobrepeso grado I
27-29,9	Sobrepeso grado II (preobesidad)
30-34,9	Obesidad de tipo I
35-39,9	Obesidad de tipo II
40-49,9	Obesidad de tipo III (mórbida)
>50	Obesidad de tipo IV (extrema)

El histograma resultante de nuestra población es la siguiente:

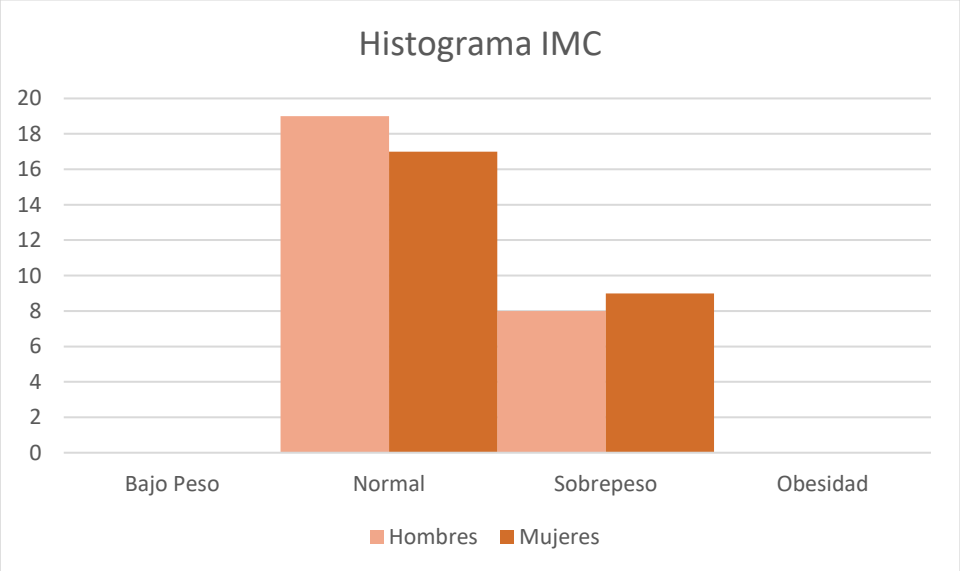


Ilustración 15. Histograma IMC de la población

Respecto al número de muestras, se ha comentado con anterioridad la existencia de dos protocolos y las muestras que se van a adquirir por cada protocolo. En la

Tabla 3, se puede observar un resumen de las muestras que se han adquirido por individuo.

Tabla 3. Número de muestras por individuo.

	PROTOCOLO A	PROTOCOLO B	TOTAL
BEMA	5 muestras	5 muestras	10 muestras
BEMC		5 muestras	5 muestras
BCMC	5 muestras	5 muestras	10 muestras
			25 muestras

Al haber reclutado 51 individuos, el número de muestras adquiridas es de 1275 muestras como se puede observar en la Tabla 4.

Tabla 4. Número total de muestras registradas por protocolo y posición

	PROTOCOLO A	PROTOCOLO B	TOTAL
BEMA	255 muestras	255 muestras	510 muestras
BEMC		255 muestras	255 muestras
BCMC	255 muestras	255 muestras	510 muestras
			1.275 muestras

4 Hardware

4.1 Arduino

Arduino es una plataforma electrónica de código abierto apoyado en hardware y software de fácil uso[21]. El hardware utilizado es un microcontrolador AT mega de arquitectura RISC. Estas placas son capaces de leer entradas desde distintos dispositivos como un sensor de humedad, la pulsación de un botón incluso mensajes provenientes de redes sociales[22] y convertirlas en una salida: encender un LED, activar un servo, guardar la información en un archivo o incluso publicar en línea.

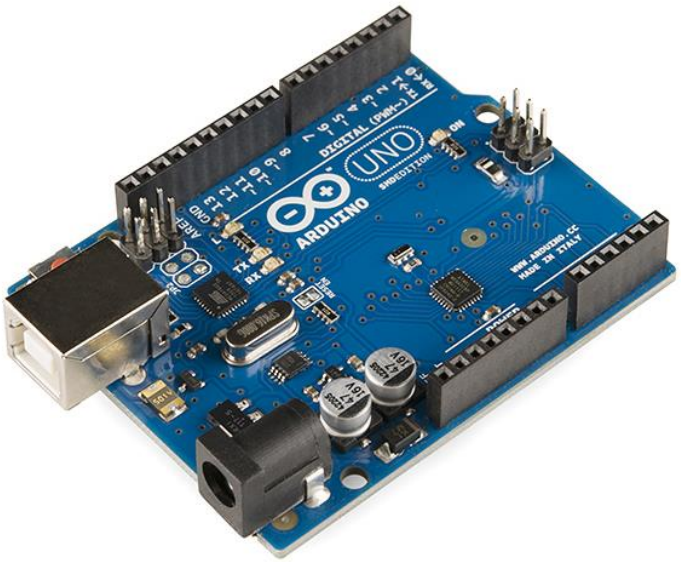


Ilustración 16. Placa Arduino Uno R3

Mandando un conjunto de instrucciones al microcontrolador de la placa, es capaz de indicarle que se desea que realice la placa.

El lenguaje de programación Arduino se divide en tres partes: funciones, valores y estructura[23] el cual se puede escribir utilizando el lenguaje C o C++ [24] y cargar en la placa mediante el software Arduino IDE del que hablaremos más adelante.

Arduino nació en el Instituto de Diseño de Interacción IVREA, como una herramienta para la creación rápida de prototipos, dirigida a un público sin experiencia en electrónica y programación. A medida que ha ido ganando terreno, Arduino ha sabido adaptarse a las nuevas necesidades diferenciando su oferta desde las tabletas de 8 bits a productos y aplicaciones para IoT, impresión portátil, impresión 3D… Arduino se ha convertido en el cerebro de miles de proyectos, desde dispositivos cotidianos hasta instrumentos científicos complejos.

4.2 Shield EMG OLIMEX

Las shields son placas de circuitos modulares que se montan sobre Arduino o sobre otro shield de forma que permite ampliar el hardware y las capacidades de Arduino[25].

Las shields se comunican con la placa Arduino ya sea por algunos de los pines digitales o analógicos o por algún bus como el SPI, I2C o puerto serie, así como usar algunos pines como interrupción. Generalmente estos Shields se alimentan a través del Arduino por los de 5V y GND.

Cada Shield de Arduino debe tener el mismo factor de forma que el standard de Arduino con un espaciado de pines concreto para que solo haya una forma posible de encajarlo.

El Shield EKG-EMG de Olimex permite capturar señales de electrocardiografía y electromiografía[26] y poder abrir nuevas posibilidades de experimentación con bio feedback. Este puede monitorizar los latidos del corazón y registrarlos o monitorizar y analizar la actividad muscular y reconocer gestos. En la propia página de Olimex viene un código de muestra que se puede descargar desde el siguiente enlace: https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG/open-source-hardware.

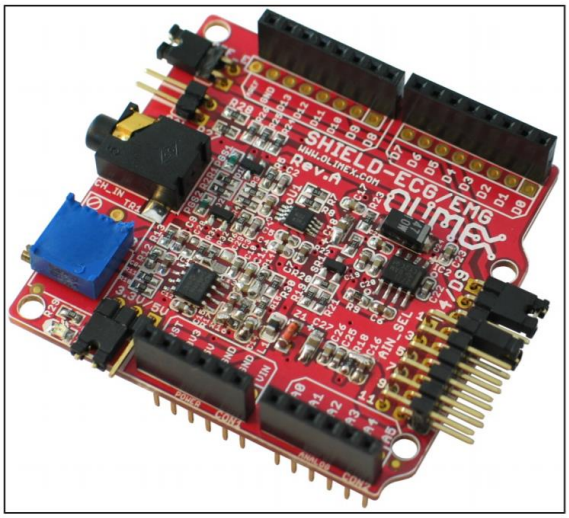


Ilustración 17. Shield EKG-EMG Olimex

El Shield EKG-EMG convierte la señal diferencial analógica (los potenciales biológicos ECG / EMG generados por los músculos), unida a sus entradas CH1_IN + / CH1_IN-, en una sola secuencia de datos como salida. La señal de salida es analógica y debe discretizarse con el objetivo de ofrecer la opción de procesamiento digital. Por lo general, esto se realiza a través de un Convertidor Analógico-Digital dedicado integrado en la MCU de la placa base (como: OLIMEXINO-328, OLIMEXINO-32U4, OLIMEXINO-STM32, etc.). En el ejemplo disponible en la página de Olimex que se ha comentado en el párrafo anterior, utiliza un ADC de 10 bits con una frecuencia de muestreo de 256Hz.

El Shield EKG-EMG necesita de unos electrodos para poder registrar la actividad necesaria. Para ello, en la misma página de Olimex se puede adquirir *Shield EKG-EMG-PA*, que se trata de un electrodo pasivo para este Shield.

Ilustración 18. Electrodos para Shield EKG-EMG de Olimex

Los electrodos están marcados como L para la mano izquierda, R para la mano derecha y D para Driven Right Leg (DRL).

5 Entorno de trabajo

5.1 Arduino IDE

El software Arduino IDE (Entorno de desarrollo integrado) de código abierto, hace que sea fácil escribir código y cargarlo en la placa. Está disponible para las plataformas más comerciales del mercado: Windows, Mac OS X y Linux. Este software permite que sea utilizado con cualquier placa de Arduino.

El código escrito para Arduino se conoce como sketch. El entorno está escrito en Java y se basa en el procesamiento, y este entorno está compuesto de las siguientes partes:

Editor de texto: aquí es donde se puede escribir el código utilizando una versión simplificada del lenguaje de programación C ++ [24].

Área de mensajes: muestra los errores y también proporciona comentarios sobre cómo guardar y exportar el código.

Texto: la consola muestra la salida de texto del entorno Arduino, incluido el error completo mensajes y otra información.

Barra de herramientas de la consola: esta barra de herramientas contiene varios botones como Verificar, Cargar, Nuevo, Abrir, Guardar y monitor en serie.

En la esquina inferior derecha de la ventana se muestra el Placa de desarrollo y puerto serie en uso.

Ilustración 19. Arduino IDE

5.2 Visual Studio

Visual Studio es un entorno de desarrollo integrado que se lanzó por primera vez en 1997 para Windows con un número reducido de lenguajes de programación soportados. Actualmente, además de soportarse en Windows, también está soportado en Linux y macOS, y es compatible con múltiples lenguajes de programación: C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, Azure, Python, Node.js, Unity…[27], [28].

Actualmente, la versión disponible es “Visual Studio 2019” y cuenta con tres ediciones: Community, Professional y Enterprise.

La versión utilizada para desarrollar este proyecto es la versión Community que es gratuita, ya que cumple los siguientes escenarios de uso admitido:

- Desarrolladores individuales
- Investigación académica

Visual Studio, desde la versión 2013 Update 1, dispone de un cliente Git integrado en el IDE[29]. Git se trata de un sistema de control de versiones distribuido y de código abierto. Además de ser gratuito es de código abierto y está diseñado para manejar desde proyectos pequeños hasta proyectos muy grandes. Gracias a la cuenta gratuita del programa “Visual Studio Dev Essentials”, se puede sincronizar el control de código Git en la propia nube de Microsoft a través de la herramienta web “Azure DevOps”, la cual proporciona servicios de desarrollador para ayudar a planificar el trabajo, colaborar en el desarrollo de código y crear aplicaciones.

5.3 Matlab

MATLAB es un programa para el computo que ejecuta gran variedad de operaciones y tareas matemáticas. Su nombre significa “MATrix LABoratory” (laboratorio de matrices) y fue diseñado en un principio para trabajar con vectores y matrices en los años 80. Actualmente abarca mucho más. Actualmente van por la versión R2019a y soportado en los sistemas operativos Microsoft Windows, Mac OS X, GNU/Linux.

El lenguaje de programación que utiliza es propio y se trata de un lenguaje interpretado, por lo que no se necesita de compilación previa de código. El código puede ejecutarse en un entorno interactivo como a través de scripts.

5.4 R Studio

R es un lenguaje pensado para la computación estadística y gráficos[30] que se lanzó por primera vez en el año 2011. A diferencia de los anteriores softwares (Matlab, Visual Studio), el motor del lenguaje y el IDE van por separado, ya que existen distintas interfaces para poder trabajar con R.

El IDE que utilizaremos en este trabajo será RStudio, ya que se trata de un IDE creado **exclusivamente** para R, y permite:

- El resaltado de sintaxis, auto completado de código y sangría inteligente.
- Ejecutar código R directamente desde el editor de código fuente.
- Salto rápido a las funciones definidas.

6 Desarrollo

En este apartado se pasa a describir todo el proceso realizado durante este trabajo para:

- Adquisición de las señales de los individuos.
- Preparación de los datos.
- Caracterización de las señales.
- Entrenamiento y evaluación del clasificador.
- Implementación del modelo en el microcontrolador.

En el apartado del diseño del trabajo se comentó que se iban a realizar dos protocolos: el protocolo A con dos tipos de posiciones y el protocolo B con tres tipos de posiciones. Debido al tiempo para la realización del trabajo, únicamente me centraré en los datos del protocolo A, y por tanto el resultado de nuestro clasificador detectará solamente dos posiciones. Para futuros trabajos, se podrán utilizar los datos del protocolo B y poder implementar un clasificador de tres posiciones del brazo y de la mano.

6.1 Adquisición señal

El primer paso del trabajo ha sido hacer funcionar un dispositivo de captura de señales de electromiografía basado en un dispositivo Arduino. Un EMG no es más que un amplificador eléctrico con varios filtros de paso alto para eliminar ruido. Si observamos el esquema del circuito del dispositivo Shield EKG-EMG de Olimex disponible en la Ilustración 20, decidí adquirir dicho dispositivo, ya que su construcción quedaba fuera del alcance de los objetivos del trabajo.

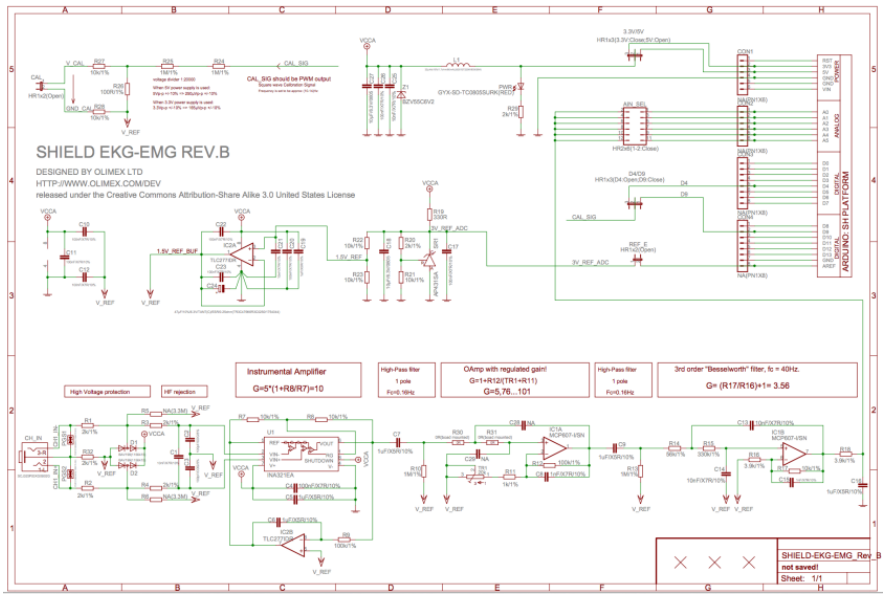


Ilustración 20. Circuito electrónico Shield EKG-EMG de Olimex

La página web del propio fabricante Olimex, únicamente proporciona: un manual escueto del Shield; Esqueña del circuito electrónico en varios formatos; Software

“Electric Guru Monitoring”; ejemplo en Arduino, Maple y Pinguino para probarlo con Electric Guru.

Lo primero que se realiza es analizar el funcionamiento del código proporcionado para el Sketch de Arduino, en el que con un poco de conocimiento de Arduino y programación se puede intuir el funcionamiento del ejemplo para poder realizar los cambios necesarios y adaptarlos al trabajo.

Las placas Arduino UNO contienen un conversor de analógico a digital multicanal de 10 bits, lo que significa que cuantificará a los voltajes de entrada entre 0 y el voltaje de operación 5V (podría ser también 3.3V) en valores enteros entre 0 y 1023.

El Shield EKG-EMG proporciona lecturas desde el pin A0 al A5, y mediante un temporizador definido en el propio Arduino[31], este recopila los valores int desde 0 a 1023 que provienen de su aparato de lectura, en nuestro caso los electrodos Shield EKG-EMG-PA, y estos se retrasmiten en cada paquete en el array data del tipo uint16_t.

Si se observa el código proporcionado por el fabricante, se puede ver la definición de un paquete de 17 bytes utilizado para transmitir información a través del puerto serial.

```
struct Olimexino328_packet
{
  uint8_t sync0;     // = 0xa5
  uint8_t sync1;     // = 0x5a
  uint8_t version;   // = 2 (packet version)
  uint8_t count;     // packet counter. Increases by 1 each packet.
  uint16_t  data[6];  // 10-bit sample (= 0 - 1023) in big endian (Motorola) format.
  uint8_t switches; // State of PD5 to PD2, in bits 3 to 0.
};
```

Ilustración 21. Estructura paquete Olimexino328_packet

En el comentario en la declaración de la variable data[6] de la estructura de datos olimexino328_packet, se observa que el tipo de la variable es un uint de 16 bits, pero realmente almacenamos 10 bits y que además, en el mismo comentario añade lo siguiente: “in big endian (Motorola) format.

Si se continúa analizando el código, sé puede ver que, por cada lectura de voltaje para cada pin, la codifica en dos bytes, un byte bajo y otro byte alto en lugar de un uint de 16 bits.

```
//Read the 6 ADC inputs and store current values in Packet
for(CurrentCh=0;CurrentCh<6;CurrentCh++){
  ADC_Value = analogRead(CurrentCh);
  TXBuf[((2*CurrentCh) + HEADERLEN)] = ((unsigned char)((ADC_Value & 0xFF00) >> 8));  // Write High Byte
  TXBuf[((2*CurrentCh) + HEADERLEN + 1)] = ((unsigned char)(ADC_Value & 0x00FF)); // Write Low Byte
}
```

Ilustración 22. Transformación Big endian

Esto es debido a que está utilizando el formato big endian nombrada anteriormente. Este formato lo que hace es representar los bytes en un orden natural, de forma que un valor hexadecimal 0x8765 se codificará como {0x87 0x65}.

Si volvemos a observar el código de la Ilustración 22 se puede ver como el primer byte de la lectura de voltaje del pin se recupera cortándolo, utilizando la operación matemática & FF00 y posteriormente desplazándose el resultado 8 bits a la derecha (>>8) de forma que se recupera el byte más significativo (MSB).

A continuación, utilizando la operación matemática & 00FF recupera el byte menos significativo (LSB). Así, manda la información en el paquete Olimexino328_packet el MSB y el LSB de forma separada.

Una vez comprendido el código proporcionado por el propio fabricante, se pasa a realizar los cambios pertinentes en el código proporcionado para realizar el propio ADC.

Se deberá definir cuál será la frecuencia de muestreo. Basándome en la teoría de Nyquist y sabiendo que el ancho de banda del bíceps (músculo estriado) se encuentra en el rango de 2-500Hz, la frecuencia mínima de muestreo deberá ser al menos de 1000Hz. Por tanto, se define la frecuencia de muestreo en **1024Hz**. Dicho de otro modo, el ADC leerá 1024 veces el valor de los electrodos conectados al bíceps.

Si nos fijamos en la Tabla 1 mostrada anteriormente, y con la definición de la frecuencia de muestreo, podemos calcular el número de valores que compondrán cada una de las muestras de 5 segundos tomadas:

5 segundos por muestra * 1024 Hz = 5.120 valores por muestra.

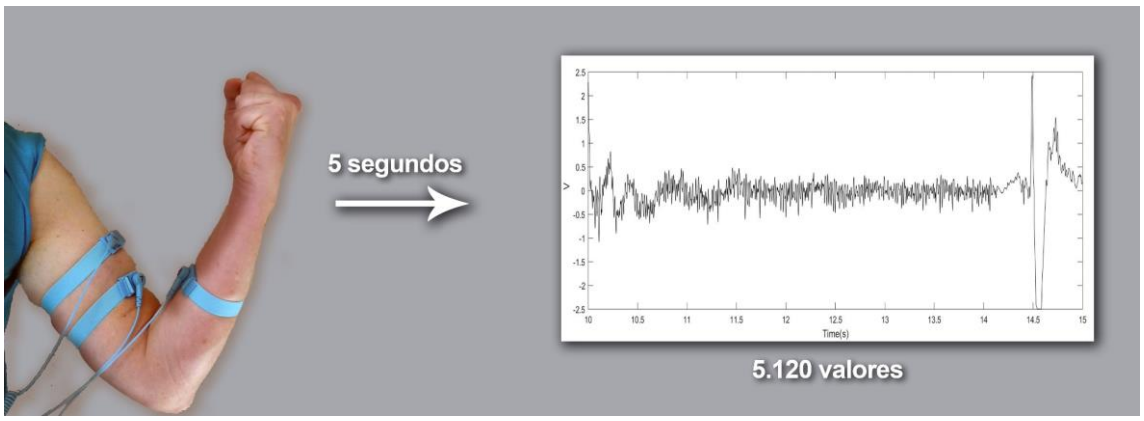


Ilustración 23. Representación de ADC muestra de 5 segundos

El número de lecturas de voltaje que se transmitirán y se almacenarán por cada uno de los registros dependiendo del protocolo serán las siguientes:

PROTOCOLO A: 60 segundos * 1024 Hz = 61.440 valores.

PROTOCOLO B: 85 segundos * 1024 Hz = 87.040 valores.

El esquema del ADC propio será:

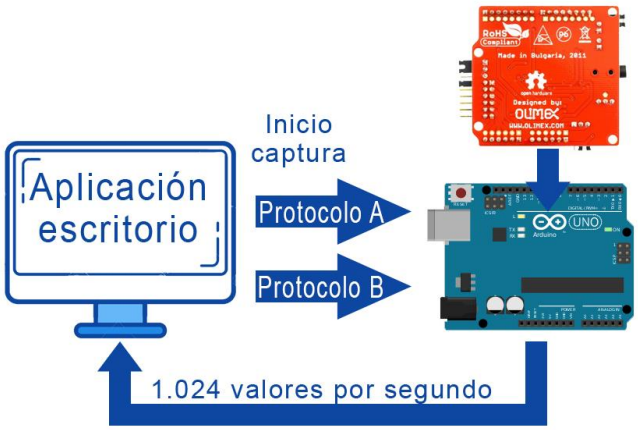
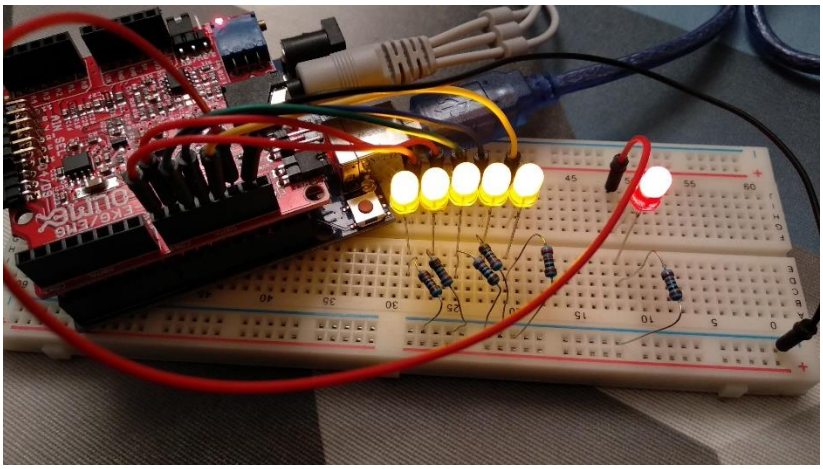


Ilustración 24. Esquema ADC propio.

Para poder indicar a los individuos los 5 segundos de duración por muestra, se construye en la placa de prototipado un conjunto de LEDS gestionados desde el propio código de Arduino para que los individuos sepan el momento exacto de cambio de posición.



Del código del fabricante, se modifica la definición del paquete a transmitir, porque únicamente se va a leer un único canal del Shield EMG y de esta forma se disminuye el tamaño del paquete a transmitir. Se añadirán dos bytes iniciales que servirán de sincronización para asegurar el inicio del paquete, y dos bytes donde se transmitirá el valor de la lectura del voltaje en ese instante del muestreo. Reduciremos en 4 Bytes la transmisión por lectura. Con la frecuencia de muestreo definida en 1024 Hz, se transmitirán:

4Bytes * 8 bits * 1024 Hz = 32.768 bps.

El Sketch para Arduino realizado para este apartado está disponible en el siguiente enlace https://github.com/ripollete/TFM_EMG/tree/master/Arduino.

Con este Sketch programado, se pasó a la tarea de adquirir señales de muestra de individuos y a almacenarse los datos de forma estructurada en la base de datos SQL Server a través de la aplicación realizada para ello.

6.2 Preparación de los datos

Una vez finalizada la etapa de la adquisición y almacenamiento de todas las muestras registradas, se deberá extraer, almacenar y organizar cada uno de estos registros en archivos CSV para poder procesar los datos mediante el software informático Matlab.

Para ello, se exporta la tabla sample_data de la base de datos SQL Server en un mismo archivo CSV, teniendo en cuenta lo siguiente: Los datos almacenados en la base de datos tienen un rango de 0 a 1023, pero se necesita los valores en voltaje. Para ello, utilizando la siguiente query, aplicamos la conversión de la cuantificación del ADC a Voltios.

```
select (A0-512)*0.0048828125,S.id_person,SD.id_sample,S.signal_type,[order]
from sample_data SD inner join sample S on SD.id_sample = S.id
order by id_sample,S.signal_type,[order]
```

La conversión hará que un valor 0 en la base de datos equivaldrá a -2,5 voltios, un valor de 512 equivaldrá a 0 voltios y un valor de 1023 equivaldrá a 2,5 voltios.

Con el archivo CSV anterior extraído directamente desde la base de datos SQL Server, se escribe un Script en R (disponible en https://github.com/ripollete/TFM_EMG/tree/master/R) donde se extraen los valores de las muestras y se organizan en carpetas.

La estructura de los archivos tendrá el siguiente formato: Protocolo(A/B)/codigoIndividuo/*codigoIndividuo_Protocolo(A/B)_idSample.csv.*

Un ejemplo de la estructura de los ficheros y carpetas sería el siguiente:

Carpeta A
- Carpeta 10001 (Datos del individuo 10001)
  - Archivo 10001_A_1248.csv
- Carpeta 10011 (Datos del individuo 10011)
  - Archivo 10011_A_1234.csv
- ….

Carpeta B
- Carpeta 10001 (Datos del individuo 10001)
  - Archivo 10001_B_1254.csv
- Carpeta 10011 (Datos del individuo 10011)
  - Archivo 10011_B_1240.csv
- ….

Estos archivos están disponibles en el siguiente enlace: https://github.com/ripollete/TFM_EMG/tree/master/CSVs .

Para analizar las distintas muestras de las señales de electromiografía, no habrá más que aislar las muestras en tramos de 5 segundos por cada registro. Pero el cambio de posición de los individuos no es exacto. Existen individuos que han cambiado de posición unas décimas de segundo antes o después, y también se

observa/detecta que el Shield EKG-EMG sufre algunas veces fluctuaciones de la señal en los cambios de posición. Por ello se debe realizar una buena segmentación y analizar cada una de las muestras de las señales así como anotar el momento de inicio y el momento de fin para tener un buen aislamiento.

Posteriormente se pasa a segmentar los registros de cada archivo CSV, generando un archivo de Script en Matlab denominado carga_registro.m (disponible en https://github.com/ripollete/TFM_EMG/tree/master/Matlab) el cual lee el archivo CSV con el identificador que se le pasa como parámetro y lo dibuja en Matlab, para de una forma visual con la herramienta de marcado anotar en una hoja de Excel el inicio y fin de cada una de las señales.

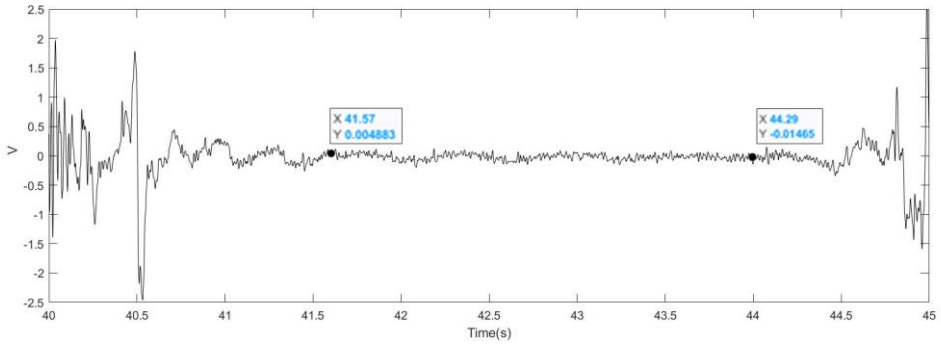


Ilustración 25. Ejemplo de segmentación de la señal.

Para anotar los datos de la segmentación se emplean dos hojas Excel, una por cada protocolo, en la que se anota el *código* del individuo, el *id* del registro y el T_{inicio} y T_{fin} de cada una de las muestras tal y como se puede observar en la Tabla 5.

Tabla 5. Estructura hoja Excel para la segmentación de los datos

Código individuo	Id del registro	Posición 1 Repetición 1 Tiempo inicio	Posición 1 Repetición 1 Tiempo fin	…..	Posición n Repetición 5 Tiempo inicio	Posición n Repetición 5 Tiempo inicio

Nota: A partir de ahora, se va a centrar únicamente con las muestras y datos del protocolo A.

Una vez segmentadas todas las muestras y con el archivo Excel rellenado, se elabora un Script en Matlab para transformar este Excel en el que cada fila contiene 10 muestras (5 muestras por cada posición) a generar un Dataset en que cada fila contenga: tipo de posición, id de la muestra, T_{inicio} y T_{fin}. El archivo se llama *protocolo_A.m* y está disponible desde este enlace: https://github.com/ripollete/TFM_EMG/tree/master/Matlab.

Los pasos que realiza este Script son los siguientes:

- Llamar a la función carga_segmentosPA: Esta función se encarga de leer el Excel que tiene la estructura de la Tabla 5 y se guarda en un Dataset

llamado datas_crudo_segmentos_PA. El archivo de esta función se llama *carga_segmentosPA.m* y está disponible en el enlace anterior.

- A continuación, el script se encarga de transformar los datos del DataSet de forma individualizada en filas. De modo que por cada fila se registrará el tipo de posición, el id del registro y el T_{inicio} y T_{fin}. La estructura resultante se guarda en el Dataset datas_A y es como sigue:

Tabla 6. Estructura inicial Dataset datas_A

tipo	Code	INI	FIN

**Si el registro del protocolo A de un individuo fueran todas las muestras correctas, de cada fila del Dataset leído del Excel se transformaría en 10 filas en el Dataset datas_A.*

El tipo está codificado con los valores 1 o 2, que equivale:

1. Brazo extendido con la mano abierta (BEMA).

2. *Brazo contraído con la mano cerrada* (BCMC).

6.3 Extracción de características

Como se ha mencionado anteriormente, existen dos técnicas de análisis de la señal, ya sea temporal o espectral.

Inicialmente, el tutor D. José Alberola me proporcionó una función para Matlab que devolvía una serie de características de la señal utilizando las dos técnicas de análisis de la señal. Investigando y realizando pruebas con la librería arduinoFFT [32] se tuvo que descartar la extracción de la técnica espectral debido a los escasos recursos tanto de procesamiento como de memoria con los que cuenta Arduino Uno.

El Toolbox para Matlab “EMG Feature Extraction Toolbox” [33], contiene funciones para extraer características de la señal basadas en el análisis temporal. De todas las disponibles del Toolbox, se eligen las siguientes 8, las cuales se basan en la media de los datos:

1. Valor absoluto medio mejorado (EMAV)
2. Valor absoluto medio (MAV)
3. Cuadrado medio raíz (RMS)
4. Cambio de amplitud promedio (AAC)
5. Diferencia del valor de desviación estándar absoluto (DASDV)
6. Valor absoluto medio modificado (MMAV)
7. Valor absoluto medio modificado 3 (MMAV2)
8. Varianza de EMG (VAR)

Con todos los datos listos en el Dataset datas_A, el Script del archivo *protocolo_A.m* continua con su ejecución y para caracterizar las señales de cada una de las filas.

Con la ayuda de un bucle For, se recorre cada una de las filas datas_A. En cada una de estas iteraciones se realiza lo siguiente:

- Se busca el archivo CSV con el código *code* de cada fila que representa una muestra segmentada, y mediante una conversión se extraen los valores entre T_{inicio} y T_{fin}. Esta conversión no es más que convertir los datos de tiempo que están expresados en segundos a cuáles de los 61.440 valores del archivo CSV corresponde. Por ejemplo, si se desea extraer desde INI = 10,14 segundos a FIN = 14,34 segundos, cogería los valores entre:
  INI = 10,14*1024 = 10.383
  FIN = 14,34*1024 = 14.684
  Esta función de extracción de la señal se encuentra en el archivo *extrae_senyal.m* disponible en la carpeta Matlab del repositorio en GitHub.
- Esta señal se pasa a la función Electro_parameter_momt (archivo electro_parameter_momt.m) que devuelve los valores de cada una de las 8 características indicadas anteriormente.
- Estos valores se añaden en columnas a datas_A, quedando la estructura del Dataset como sigue:

Tabla 7. Estructura Dataset datas_A con características

tipo	code	INI	FIN	Carac. 1	Carac. 2	….	Carac. 8

Con lo aprendido en la asignatura cursada de Machine Learning, los datos fisicoquímicos a la hora de utilizarlos para algoritmos de clasificación funcionan mejor si estos están normalizados entre 0 y 1, y mediante la siguiente función disponible en el archivo *normalizacion.m,* normalizo los valores:

```
function norm = normalizacion_v1(datas_)
    norm =[];
    datas_= table2array(datas_);
    max_norm = max(datas_);
    min_norm = min(datas_);
    for i = 1:size(datas_,1)
        x = datas_(i);
        norm(i) = (x - min_norm) / (max_norm - min_norm);
    end
    norm = table(transpose (norm));
end
```

En el Anexo 2, se añade los Boxplots resultantes de cada una de estas características agrupadas por tipo de posición del brazo y de la mano (posición 1 o 2).

6.4 Entrenamiento del clasificador

La función *fitcdiscr(Tbl,Y)* de Matlab nos devuelve un modelo de análisis discriminante ajustado basado en las variables de entrada, o también llamadas predictoras, características o atributos, contenidos en la tabla Tbl y las respuestas de estas en el array Y.

La primera decisión que hay que tomar es la división de los datos recogidos para seleccionar cuales van a ser los datos de entrenamiento y cuales lo de prueba tal y como se ha visto en la asignatura de Machine Learning [15].

A continuación, se deberá obtener el mejor modelo seleccionando las características de la señal que mejor resultado nos dé. Uno de los estimadores para poder visualizar la eficacia de los modelos de aprendizaje automático es el Área bajo la curva (AUC). La curva de Característica operativa del receptor (ROC) se utiliza para examinar la compensación entre la detección de verdaderos positivos mientras evita los falsos positivos.

6.4.1 División de los datos

La teoría de la asignatura de Machine Learning nos suele indicar en la división de datos con la proporción: 70% para entrenar el modelo y 30% para realizar el test. Debido a la cantidad de datos de los que se disponen para el protocolo A, se toma la decisión de dividir los datos en tres grupos: datos para entrenar el modelo, datos para hacer el test del modelo y datos de validación que siempre serán los mismos para validar todos los modelos resultantes.

La proporción será:

Tabla 8. Proporcion datos train, test y validation

<table>
<tr><td>20%</td><td colspan="2">Datos para validación</td></tr>
<tr><td rowspan="2">80%</td><td>70%</td><td>Datos para entrenamiento</td></tr>
<tr><td>30%</td><td>Datos para test</td></tr>
</table>

Para realizar una partición de los datos de forma aleatoria y reproducible obteniendo los mismos resultados utilizaré una semilla inicial y un Step que sumaré cada vez que se realice una iteración:

```
seed = 816225;
seed_step = 981365;
```

Estos valores han sido asignados la primera vez ejecutando la función *randsample(1000000,1)*.

6.4.2 Elección de las características

En un primer momento se realizan pruebas utilizando la función *sequentialfs* de Matlab para la elección automática de características, utilizando como he comentado con anterioridad el uso de semillas para obtener siempre los mismos resultados.

Como indica la documentación de sequentialfs: “inmodel = Sequentialfs(fun,X,y selecciona un subconjunto de entidades de la matriz de datos que mejor predice los datos mediante la selección secuencial de entidades hasta que no haya ninguna mejora en la predicción. Las Filas *X* se corresponden a las observaciones; las columnas se corresponden a variables o características. *y* es un vector de columna de valores de respuesta o etiquetas de clase para cada observación en *X*. *X e y* debe tener el mismo número de filas. *fun* es un identificador de función para una función que define el criterio utilizado para seleccionar entidades y determinar cuándo detenerse. La salida *inmodel* es un vector lógico que indica qué entidades se eligen finalmente” [34] .

El funcionamiento de esta función es secuencial: “A partir de un conjunto de entidades vacío, crea subconjuntos de entidades candidatas añadiendo secuencialmente cada una de las entidades aún no seleccionadas. Para cada subconjunto de características candidatas, *sequentialfs* realiza una validación cruzada de 10 veces al llamar *fun* repetidamente con diferentes subconjuntos de entrenamiento de *X* e *y, XTRAIN and* ytrain, y los subconjuntos de prueba *X* e *y, XTEST* and *ytest*”.

Esta función admite parámetros para poder configurar por ejemplo la dirección en el que irá evaluando las características, si *“forward”* o *“backward”* al igual que el número de variables que se quiere escoger. Pero se observa que la elección de las características, aunque siempre nos devolverá las que más puntuación se obtenga, siempre dependerá del orden inicial en que estén dispuestas las características de la señal.

Es por ello que se realiza un Script que ejecute todas las combinaciones posibles de las características disponibles. Este archivo se llama *Entrenamiento_combinaciones.m* y está disponible en el repositorio dentro de la carpeta Matlab. El Script seleccionará desde 2 combinaciones de características hasta 8 combinaciones de características. Por cada una de estas combinaciones se obtendrá 250 modelos de forma aleatoria y se validará con la función predict con tres conjuntos de datos: Los que se utilicen para el entrenamiento, los que se hayan definido para el test, que cada vez serán distintos y con los datos de validación que siempre serán los mismos.

```
datos_validacion<- 20% datas_A

datos_bucle <- 80% datas_A

Semilla <- establecer semilla

Step <- establecer step semilla

Repetir para todas las combinaciones posibles de características

      Repetir 250 veces

            Semilla <- Semilla+step

            datos__entrenamiento <- 70% datos_bucle

            datos_test <- 30% datos_bucle

            Mdl <- Modelo fitcdiscr() con datos_entrenamiento

            AUC1   <-   Obtener   AUC   mediante   predict   con   los
datos_entrenamiento

            AUC2 <- Obtener AUC mediante predict con los datos_test

            AUC3 <- Obtener AUC mediante predict con los datos_validacion

            MediaAUC <-Obtener desviación AUC1, AUC2 y AUC3

            Desviación <- Obtener desviación AUC1, AUC2 y AUC3

          Almacenar en una tabla MediaAUC, Desviación y datos de la
     iteración.

      FIN

      Almacenar en una tabla la media de las 250 iteraciones anteriores
y las variables que se han utilizado para dichos valores.

FIN
```

El resultado de este Script se puede ver en el Anexo 3, en el que se puede observar que la mejor combinación corresponde a la combinación de las 4 características: MAV, DASDV, MMAV y MMAV2 obteniendo un valor medio **AUC de 0,84078** y con una **desviación del 0,0241444**.

Una vez decidido que se va a obtener el modelo con estas cuatro características, se vuelve a ejecutar el bucle interior de 250 iteraciones con el Script del archivo *entrenamiento_A.m* pasándole únicamente estas 4 características, y obteniendo como resultado una tabla que contendrá la media de cada una de las iteraciones y sus coeficientes para del modelo.

La combinación de decisiones de varios clasificadores puede llevar a obtener un mejor resultado en la decisión [35]. La aplicación del voto mayoritario (majority voting) es una estrategia simple y tan efectiva como esquemas complicados utilizados para la mejora de los resultados.

Los 9 coeficientes de los mejores resultados son los siguientes:

Tabla 9. 9 mejores coeficientes DLA con 4 características.

Coeff 1	Coeff 2	Coeff 3	Coeff 4	Coeff 5	AUC
-55,8	-50,929	113,14	-59,205	1,9511	0,8645
-75,108	-46,645	134,14	-57,702	2,2369	0,8642
-59,402	-48,104	116,66	-57,468	1,8863	0,8635
-62,18	-44,898	110,6	-47,566	1,9991	0,8635
-77,386	-45,448	140,99	-60,533	2,0675	0,8633
-79,672	-54,29	155,14	-75,31	2,2665	0,8616
-82,112	-51,816	152,75	-66,436	2,13	0,8611
-77,623	-44,576	140,64	-60,739	2,0248	0,8601
-73,872	-50,028	139,29	-61,479	1,9156	0,86

Al tratarse de un modelo DLA, la ecuación resultante de cada uno de los modelos será:

$$Y = Coeff1 * MAV + Coeff2 * DASDV + Coeff3 * MMAV + Coeff4 * MMAV2 + Coeff5$$

Ecuación 2

Si Y < 0 se clasificará como tipo 1, sino se clasificará como tipo 2.

6.5 Implementación en el microcontrolador

Ya que la parte de la adquisición de la señal y el Timer ya está implementada de la primera etapa de la adquisición de la señal al construir el ADC con el Shield EKG-EMG de Olimex, únicamente nos centraremos en la parte de la programación de la caracterización de la señal, construcción del modelo y la estrategia del voto mayoritario.

Lo primero es la creación de una librería en Arduino para poder extraer en el propio Arduino las características de una señal dada como las utilizadas en el Toolbox de Matlab “EMG Feature Extraction Toolbox”. Aunque las definiciones de todas las características están en multitud de bibliografías, se intenta reproducir lo más fielmente las funciones de este Toolbox en el lenguaje C++ para poder crear la librería y sus funciones. La librería se llama Features_EMG y está disponible en el siguiente enlace: https://github.com/ripollete/TFM_EMG/tree/master/Arduino/Features_EMG.

En esta librería se definen dos constantes por cada una de las características, en las que se carga el valor mínimo y máximo utilizado para cada una de estas características en la parte de la normalización, y obtener una normalización idéntica que la calculada en Matlab. De este modo por cada característica tendremos dos funciones, una que devuelve el valor sin normalizar y otra que devuelve el valor normalizado entre 0 y 1.

Para comprobar que daba los mismos resultados, simule una onda sinusoidal y pasé los mismos trozos de señal tanto en Matlab con el Toolbox “EMG Feature

Extraction Toolbox" como en el propio Arduino utilizando la librería Features_EMG creada para este proyecto.

Para el almacenamiento de los valores capturados por el ADC a una frecuencia de muestreo de 1024Hz como se definió en el apartado "6.1 Adquisición señal", se utilizará un Buffer circular de 150 posiciones. Además, se utilizará un Buffer auxiliar en el que cada 150 lecturas del ADC almacenará el contenido del Buffer circular. El contenido de este Buffer auxiliar será el que se le pasará a la librería Features_EMG para la extracción de las 4 características seleccionadas.

Debido a que Arduino Uno cuenta con 2Kb de memoria SRam, la elección del tamaño del Buffer no ha sido trivial, ya que el tamaño de estos dos vectores no puede superar dicha memoria. El tipo de variable *Double*, que es la utilizada en los dos Buffers, por definición en Arduino Uno, ocupa un espacio de memoria de 4 bytes (32 bit). Si multiplicamos por 300 posiciones entre los dos Buffers, estamos utilizando 1.200 bytes, casi el 60% de su capacidad únicamente con estos dos Buffers, es por ello que no se les puede dar más tamaño.

Una vez calculadas las características con la librería Features_EMG, únicamente se debe multiplicar estos valores por los coeficientes de cada uno de los 9 modelos seleccionados y utilizando la estrategia del mayor votado y devolver cual es la decisión mayoritaria entre los 9 modelos. Los coeficientes de estos modelos se declaran al principio del código Arduino como una matriz 9x5, para facilitar su cálculo y reducir el número de líneas de código. La función utilizada para el clasificador es la siguiente:

```
int clasificador(double F1, double F2, double F3, double F4)
{
  int num_model = length(Coeffs);
  //Inicialización de las variables para la estrategia Majority voting
  int tipo_A = 0;
  int tipo_B = 0;
  //Bucle en el calcularemos el resultado de la ecuación de coeficientes
  //de los n modelos seleccionados para el Majority voting
  for (int i = 0; i < num_model; i++)
  {
    double aux = F1 * Coeffs[i][0] + F2 * Coeffs[i][1] + F3 * Coeffs[i][2]
    + F4 * Coeffs[i][3] + Coeffs[i][4];
    //Si el resultado es menor de 0 tipo_B++
    if (aux < 0)
      tipo_B++;
    //Sino, tipo_A++
    else
      tipo_A++;
  }
  //Aplicamos Majority voting
  if (tipo_A > tipo_B)
    return 1;
  else
    return 2;
}
```

El archivo resultante del clasificador se puede obtener en el siguiente enlace: https://github.com/ripollete/TFM_EMG/tree/master/Arduino.

7 Resultados

El resultado obtenido del trabajo es un dispositivo clasificador hombre-máquina que consta de un ADC para la captación de señales de electromiografía proveniente del bíceps. Caracterizando la señal por mediación de la librería Features_EMG en el propio dispositivo y pasando estos valores como parámetros de los modelos DLA obtenidos tras un entrenamiento del modelo es capaz de con un valor medio teórico AUC de **0,84078** responder a la pregunta de cuál de las dos posiciones (BEMA y BCMC) se encuentra el individuo.

La forma de indicar al usuario el resultado de la clasificación se realiza por dos medios.

- De forma visual mediante dos Leds de distinto color donde cada uno nos indica una posición distinta.
- Mediante el cable serial, el cual manda en forma de texto "*Brazo extendido - Mano abierta*" o "*Brazo contraído – Mano cerrada*"

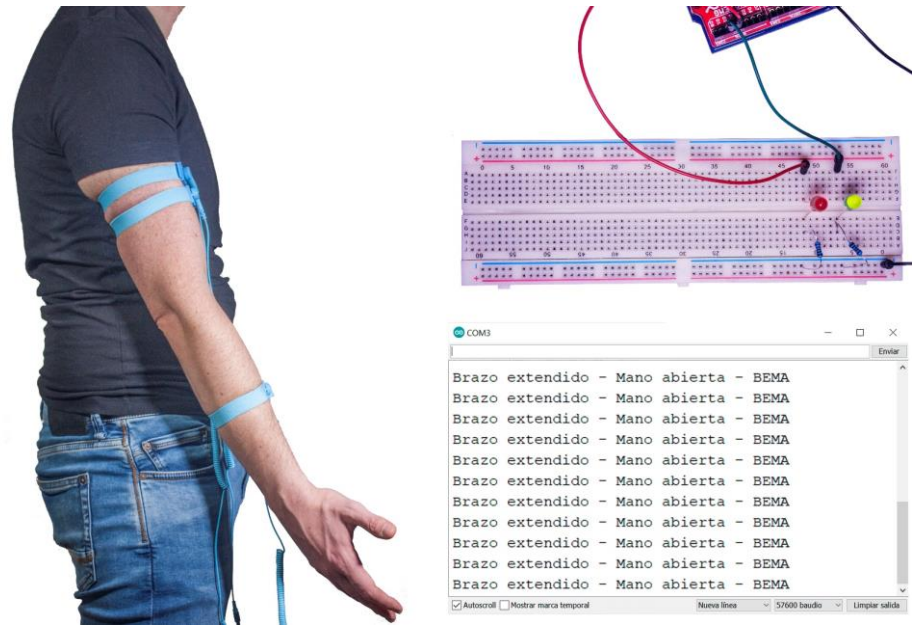


Ilustración 26. Resultado posición BEMA

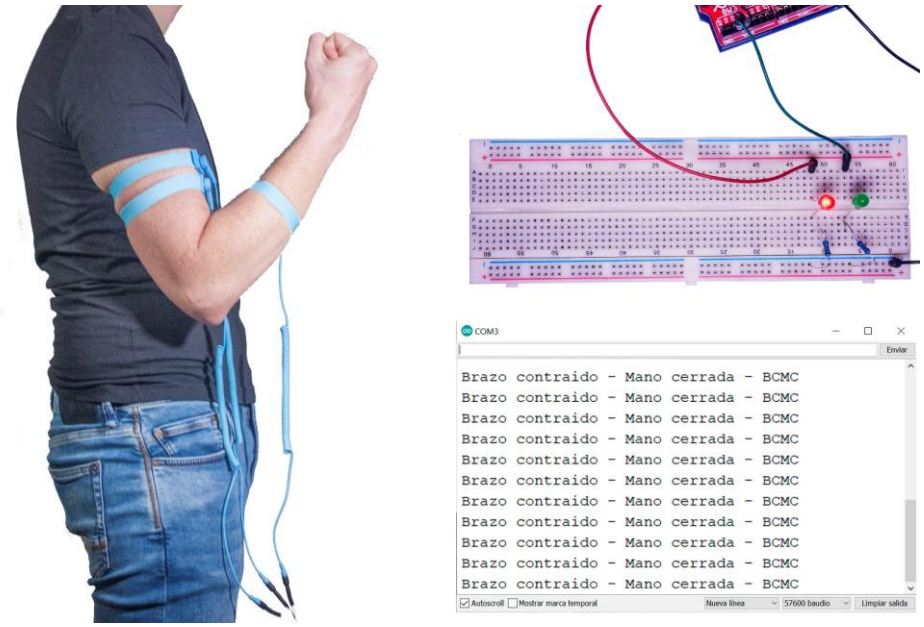


Ilustración 27. Resultado posición MCMC

El resultado del prototipo se presenta sin compactar en alguna caja. Únicamente se presenta con el Shield AKG-EMG de Olimex montado modularmente en la placa Arduino Uno, los cables de los electrodos Shield AKG-EMG AP conectados al Shield de Olimex y tres cables conectados a la breadboard. Uno de los cables irá al GND y los otros dos D2 y D3 conectados a dos bombillas Led tal y como se puede apreciar en la Ilustración 6 mostrada anteriormente en el apartado Breve sumario de productos obtenidos de la página 6.

Según lo aprendido en la asignatura de Machine Learning, el valor AUC obtenido se puede interpretar como si de una escala académica se tratara como se muestra en la tabla siguiente:

Tabla 10. Puntuación académica interpretación puntuación AUC

A	Outstanding = 0.9 to 1.0
B	Excellent/good = 0.8 to 0.9
C	Acceptable/fair = 0.7 to 0.8
D	Poor = 0.6 to 0.7
E	No discrimination = 0.5 to 0.6

El resultado del modelo obtenido tendríamos una puntuación de:

B “Excellent/Good”

8 Valoración económica

El coste del material para la realización del trabajo ha sido la siguiente:

- Kit Básico para Arduino con UNO R3. El cual está compuesto por la placa Arduino UNO R3, un Breadboard, y distintos componentes como: sensores luz, leds, pulsadores, cables, potenciómetro…
  Precio aproximado: 30€.
- Shield EKG-EMG de Olimex.
  Precio aproximado: 20€.
- Shield EKG-EMG PA de Olimex
  Precio aproximado: 10€.

Coste total de material: 60€.

9 Conclusiones

En este trabajo, he adquirido conocimientos que han permitido el aprendizaje de:

- Metodología para la adquisición de Bioseñales construyendo un ADC.
- Que son y como tratar las señales de electromiografía.
- Caracterización de Bioseñales basadas en el análisis temporal y espectral.
- Funcionamiento y extracción de los coeficientes del algoritmo DLA de Fisher.
- Encapsulación y estrategias de decisión en los modelos.

Los objetivos del trabajo y específicos se han cumplido a lo largo de todo el trabajo.

Aunque satisfecho por el resultado obtenido explicado en el apartado “7 Resultados”, no ha sido el deseado respecto a la planificación inicial por los siguientes motivos:

- El número de individuos registrados ha sido de 51, cuando inicialmente se preveía el registro de 100 individuos. Como se explicó en una de las pruebas de evaluación, el tiempo medio de registro y segmentación por individuo era de unos 26 minutos, y esto hizo que sufriera retrasos en el trabajo. El detalle de cada uno de los pasos en tiempos es el siguiente:
    - Explicación resumida del TFM al individuo. Preparar los electrodos en el brazo. Conectar los cables. Tomar sus datos: Nombre, apellidos, fecha de nacimiento, sexo, peso y talla. Explicación de los dos tipos de protocolo a seguir. *5 minutos*.
    - Realización de dos pruebas de 10 segundos cada una para comprobar de forma visual que está adquiriendo una señal deseada. *2 minutos*.
    - Protocolo A (60 segundos) y comprobación visual que la señal es correcta. *2 minutos*.
    - Protocolo B (85 segundos) y comprobación visual que la señal es correcta. *3 minutos*.
    - Exportación de las mediciones a un archivo CSV y posterior codificación en la carpeta correspondiente. *4 minutos*.
    - Segmentar cada uno de los dos registros de cada individuo. *10 minutos.*
- Otro de los logros que no se han alcanzado es la caracterización de la señal mediante análisis espectral debido a las prestaciones del microcontrolador ATmega328 y de memoria con la que dispone Arduino Uno. Es por ello que se ha tenido que realizar la caracterización basado en el análisis temporal. Las pruebas con la librería ArduinoFFT para Arduino para la caracterización basada en el análisis espectral también me han hecho sufrir un retraso importante.
- Como se indicó en el apartado del diseño, se registraron dos protocolos, para poder realizar dos clasificadores, uno de dos posiciones y otro de tres posiciones. Debido al tiempo, únicamente se ha desarrollado el clasificador para el protocolo A.

9.1 Líneas futuras

- Separación del resultado final en dos sistemas independientes. Uno para el ADC y otro para el clasificador. Además, elección de placas con controladores más potentes que las que cuenta Arduino, ya que los chips ATmega328 solamente tienen una velocidad de 16Mhz.
- Teniendo los datos disponibles y preparados, se podría realizar un clasificador para el protocolo B, el cual clasificaría entre tres posiciones del brazo y de la mano.
- Actualización de la librería Features_EMG con más características de la señal basadas en el análisis espectral y temporal para posterior publicación en repositorios públicos.
- Obtención de un brazo biónico y conectarlo al clasificador para que duplique el movimiento.

10 Glosario

ADC

Conversor Analógico-Digital

Dataset

Conjunto de datos

DSP

Procesador digital de señal (Digital signal processor)

EMG

Electromiograma

FFT

Transformada rápida de Fourier

IDE

Entorno de desarrollo integrado

IoT

Internet de las cosas (Internet of things)

LDA / ADL

Linear discriminant analysis / Análisis discriminante lineal

LDR

Resistor dependiente de la luz (Light Dependent Resistor)

MCU

Microcontrolador (Microcontroller unit)

PAC

Proves d'evaluació continua

PEC

Pruebas de evaluación continua

TFM

Trabajo final de máster

11 Bibliografía

[1] G. A. BETANCOURT, E. G. Suárez, and J. F. Franco, “Reconocimiento de patrones de movimiento a partir de señales electromiográficas,” *Scientia et technica*, vol. 10, no. 26, pp. 53–58, 2004.

[2] K. A. Farry, I. D. Walker, and R. G. Baraniuk, “Myoelectric teleoperation of a complex robotic hand,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 775–788, 1996.

[3] S. Ferguson and G. R. Dunlop, “Grasp recognition from myoelectric signals,” in *Proceedings of the Australasian Conference on Robotics and Automation, Auckland, New Zealand*, 2002, vol. 1.

[4] E. Kaniusas, “Biomedical Signals and Sensors I,” *Biomedical Signals and Sensors I*, pp. 1–21, 2012.

[5] F. Tedin and J. Fraire, “Procesamiento de Bioseñales en Tiempo Real en Universos Interactivos,” 2017.

[6] J. F. Guerrero Martínez and J. F. G. Martínez, *Ingeniería biomédica*. Moliner-40, 2004.

[7] C. A. Alva Coras, “Procesamiento de señales de electromiografía superficial para la detección de movimiento de dos dedos de la mano,” 2012.

[8] J. H. T. Viitasalo and P. v Komi, “Signal characteristics of EMG during fatigue,” *European Journal of Applied Physiology and Occupational Physiology*, vol. 37, no. 2, pp. 111–121, 1977.

[9] C. J. de Luca, “The use of surface electromyography in biomechanics,” *Journal of applied biomechanics*, vol. 13, no. 2, pp. 135–163, 1997.

[10] P. v Komi and P. Tesch, “EMG frequency spectrum, muscle structure, and fatigue during dynamic contractions in man,” *European journal of applied physiology and occupational physiology*, vol. 42, no. 1, pp. 41–50, 1979.

[11] R. Merletti and P. di Torino, “Standards for reporting EMG data,” *J Electromyogr Kinesiol*, vol. 9, no. 1, pp. 3–4, 1999.

[12] A. van Boxtel, “Optimal signal bandwidth for the recording of surface EMG activity of facial, jaw, oral, and neck muscles,” *Psychophysiology*, vol. 38, no. 1, pp. 22–34, 2001.

[13] J. Ignacio, “Conversores Análogo-Digital y Digital-Análogo: Conceptos Básicos.”

[14] M. Semeria, “Los tres teoremas: Fourier - Nyquist - Shannon,” Universidad del Centro de Estudios Macroeconómicos de Argentina (UCEMA), Buenos Aires, 2015.

[15] B. Lantz, *Machine learning with R*. Packt Publishing Ltd, 2013.

[16] F. Parra, “Estadística y Machine Learning con R,” *RPubs Blog*, 2017.

[17] “Linear discriminant analysis - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Linear_discriminant_analysis. [Accessed: 30-Dec-2019].

[18] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[19] C. Games, W. J. Lobato, S. Member, and S. S. Advisor, “Study of EMG Signals for Movements Recognition applied for controlling Study of EMG Signals for Movements Recognition applied for controlling Computer Games,” no. June, 2015.

[20] C. Gómez Piqueras, “Disociación/anonimización de los datos de salud,” *Derecho y Salud*, vol. 18, no. 1, pp. 43–49, 2009.

[21] “Arduino - Introduction.” [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Accessed: 13-Dec-2019].

[22] A. Alfonso Patiño and T. Eléctrico, “Diseño y elaboración de la guía para sistemas digitales con arduino Uno R3,” 2014.

[23] “Language Reference.” [Online]. Available: https://www.arduino.cc/reference/en/. [Accessed: 13-Dec-2019].

[24] “WORKING PRINCIPLE OF ARDUINO AND USING IT AS A TOOL FOR STUDY AND RESEARCH,” 2018.

[25] “Shields para Arduino | Aprendiendo Arduino.” [Online]. Available: https://aprendiendoarduino.wordpress.com/2015/03/23/shields-para-arduino/. [Accessed: 17-Dec-2019].

[26] “SHIELD-EKG-EMG - Open Source Hardware Board.” [Online]. Available: https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG/open-source-hardware. [Accessed: 17-Dec-2019].

[27] “Microsoft Visual Studio - Wikipedia, la enciclopedia libre.” [Online]. Available: https://es.wikipedia.org/wiki/Microsoft_Visual_Studio. [Accessed: 17-Dec-2019].

[28] “Visual Studio 2019 | Visual Studio.” [Online]. Available: https://visualstudio.microsoft.com/es/vs/. [Accessed: 17-Dec-2019].

[29] “Git - Git en Visual Studio.” [Online]. Available: https://git-scm.com/book/es/v2/Git-en-otros-entornos-Git-en-Visual-Studio. [Accessed: 17-Dec-2019].

[30] “R: What is R?” [Online]. Available: https://www.r-project.org/about.html. [Accessed: 17-Dec-2019].

[31] “Arduino Playground - FlexiTimer2.” [Online]. Available: https://playground.arduino.cc/Main/FlexiTimer2/. [Accessed: 23-Dec-2019].

[32] “ArduinoFFT - Open Music Labs Wiki.” [Online]. Available: http://wiki.openmusiclabs.com/wiki/ArduinoFFT. [Accessed: 01-Jan-2020].

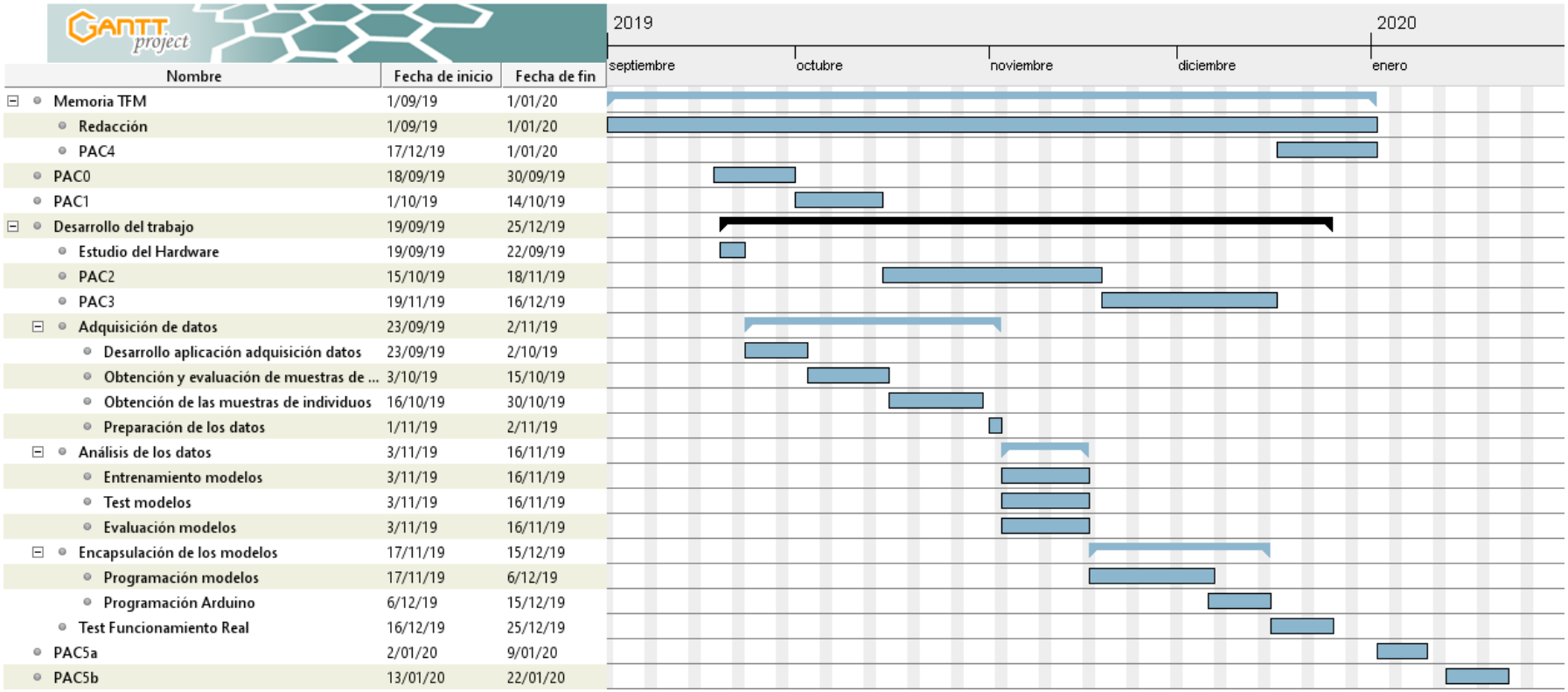
[33] Jingwei Too, “EMG Feature Extraction Toolbox,” *MATLAB Central File Exchange*, 2020. [Online]. Available: https://es.mathworks.com/matlabcentral/fileexchange/71514-emg-feature-extraction-toolbox. [Accessed: 01-Jan-2020].

[34] “Sequential feature selection using custom criterion - MATLAB sequentialfs - MathWorks.” [Online]. Available: https://es.mathworks.com/help/stats/sequentialfs.html?lang=en. [Accessed: 02-Jan-2020].

[35] L. Lam and C. Y. Suen, “Application of majority voting to pattern recognition: An analysis of its behavior and performance,” *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, vol. 27, no. 5, pp. 553–568, 1997.

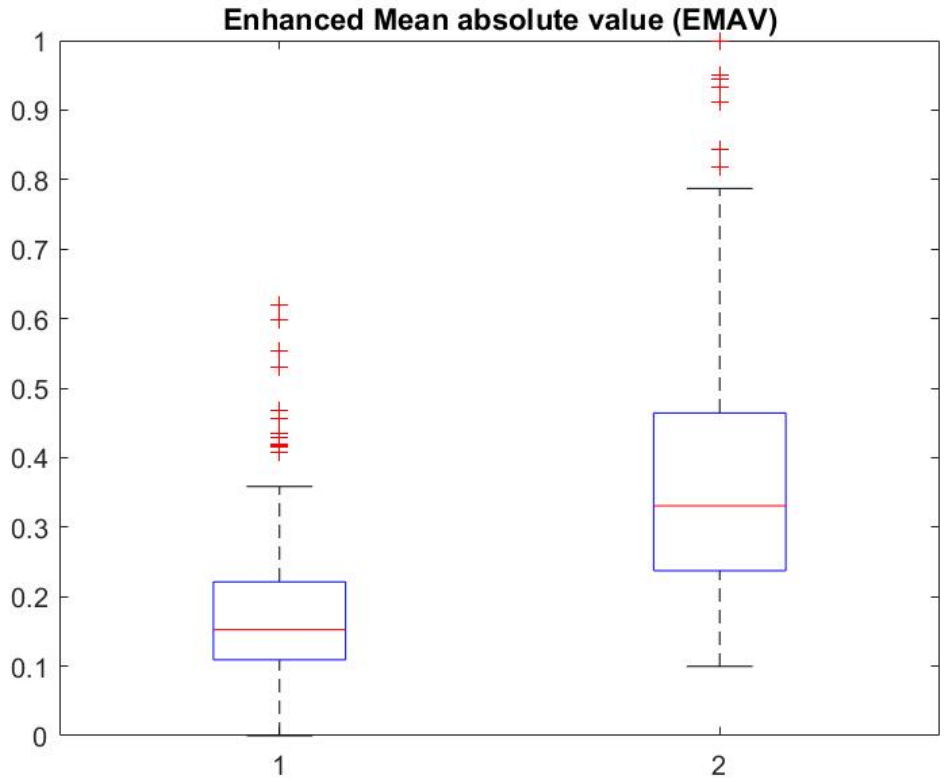
12 Anexos

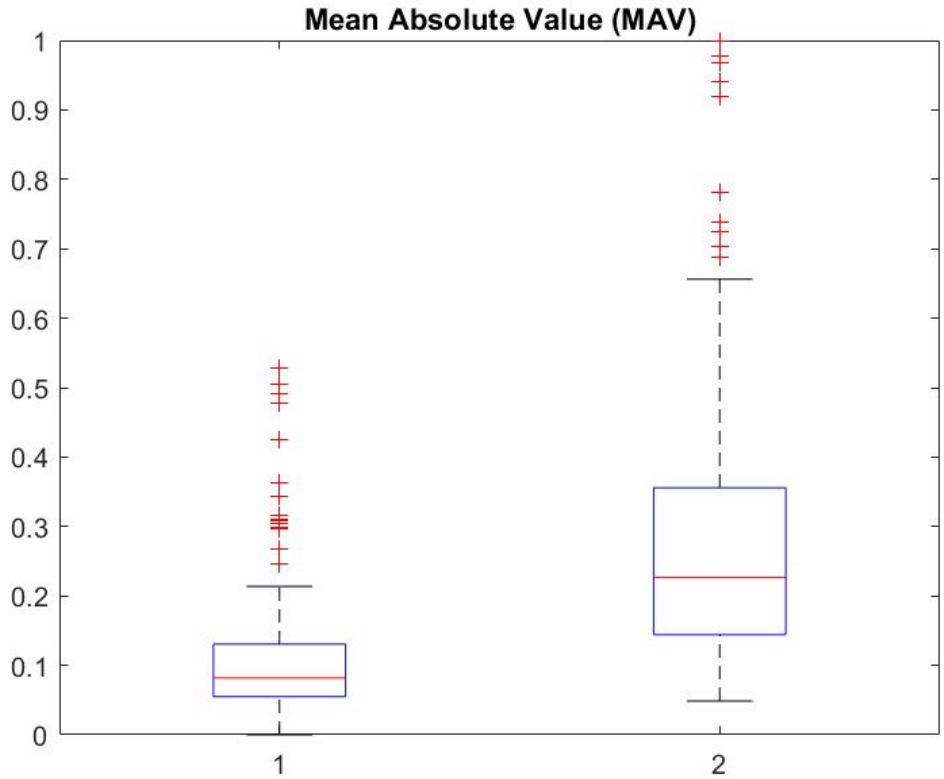
12.1 Anexo 1. Planificación del trabajo – Diagrama de Gantt

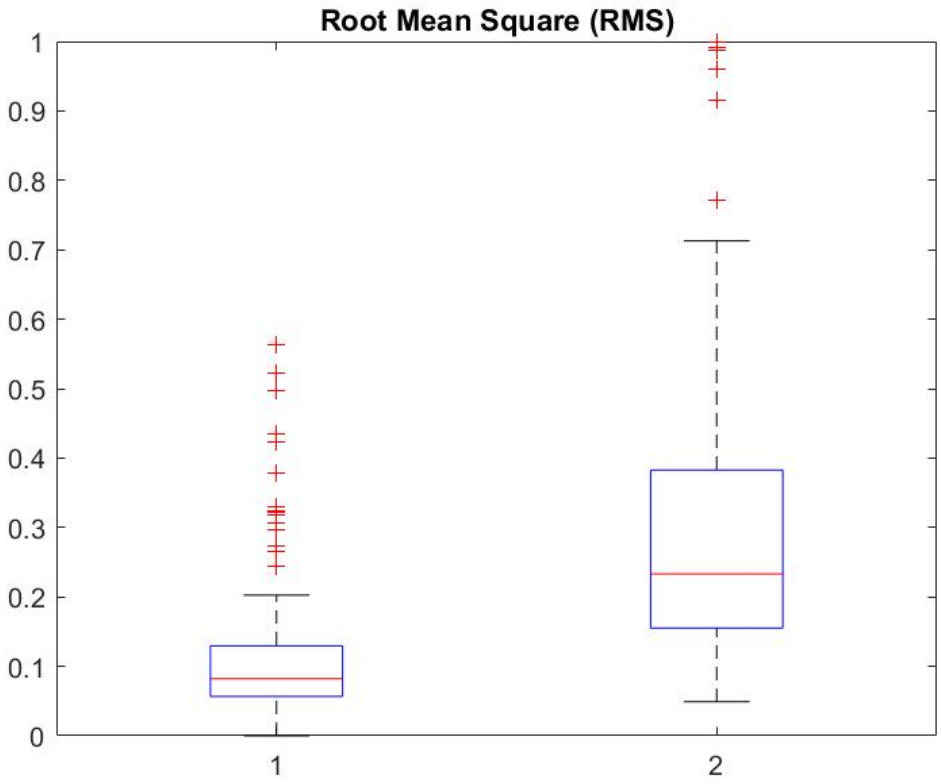


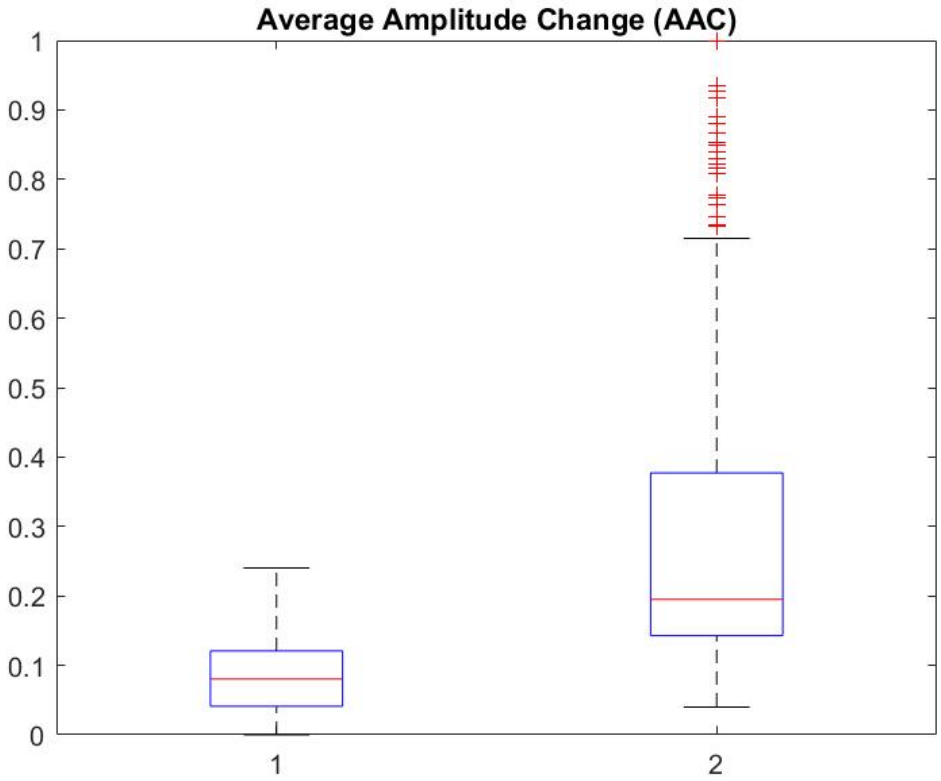
Nombre	Fecha de inicio	Fecha de fin
Memoria TFM	1/09/19	1/01/20
Redacción	1/09/19	1/01/20
PAC4	17/12/19	1/01/20
PAC0	18/09/19	30/09/19
PAC1	1/10/19	14/10/19
Desarrollo del trabajo	19/09/19	25/12/19
Estudio del Hardware	19/09/19	22/09/19
PAC2	15/10/19	18/11/19
PAC3	19/11/19	16/12/19
Adquisición de datos	23/09/19	2/11/19
Desarrollo aplicación adquisición datos	23/09/19	2/10/19
Obtención y evaluación de muestras de ...	3/10/19	15/10/19
Obtención de las muestras de individuos	16/10/19	30/10/19
Preparación de los datos	1/11/19	2/11/19
Análisis de los datos	3/11/19	16/11/19
Entrenamiento modelos	3/11/19	16/11/19
Test modelos	3/11/19	16/11/19
Evaluación modelos	3/11/19	16/11/19
Encapsulación de los modelos	17/11/19	15/12/19
Programación modelos	17/11/19	6/12/19
Programación Arduino	6/12/19	15/12/19
Test Funcionamiento Real	16/12/19	25/12/19
PAC5a	2/01/20	9/01/20
PAC5b	13/01/20	22/01/20

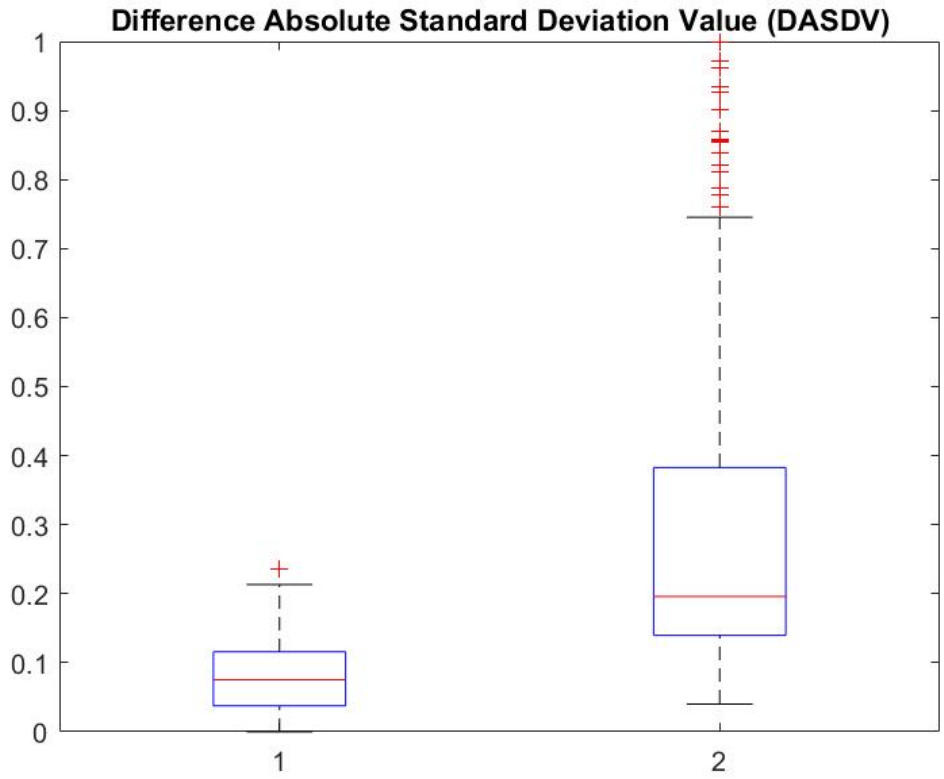
12.2 Anexo 2. Boxplots características de la señal normalizada

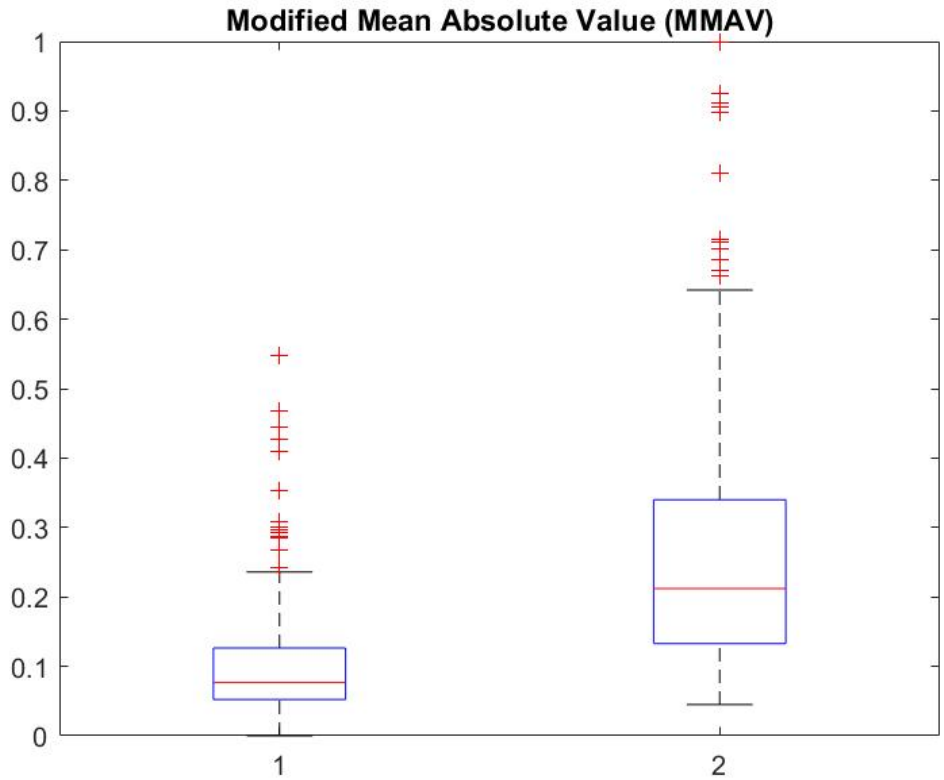


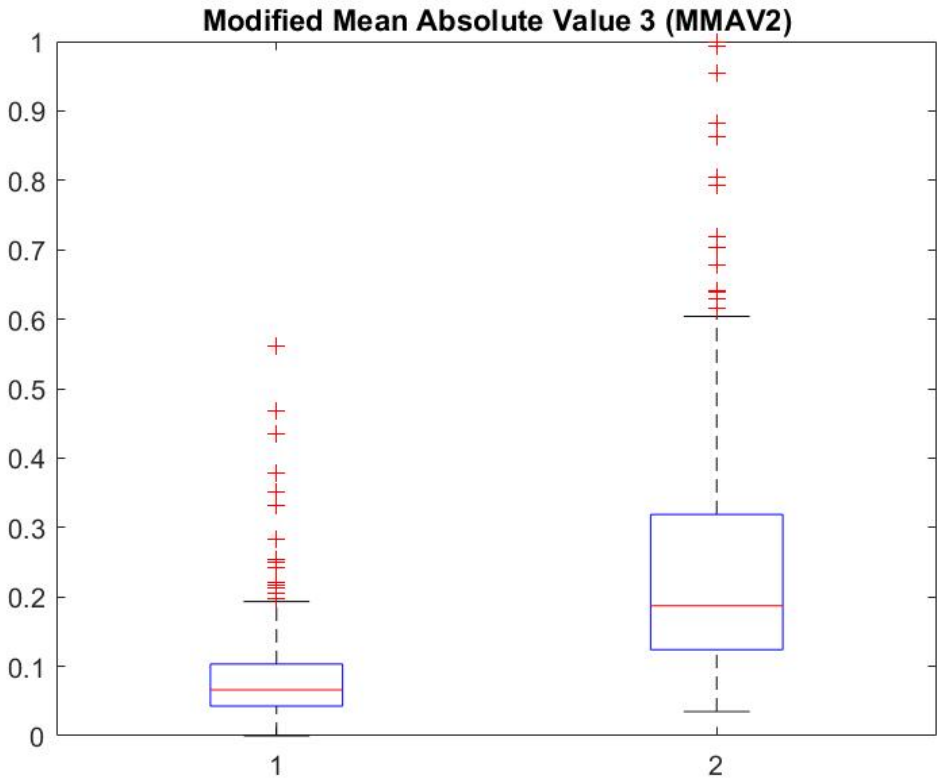


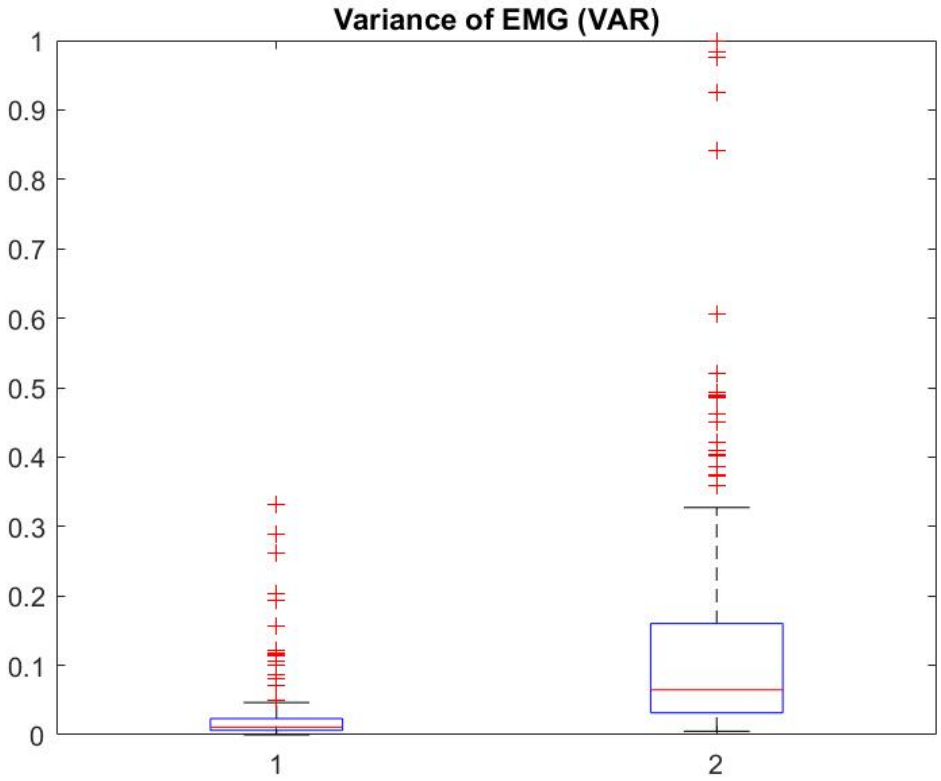

Root Mean Square (RMS)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2


Average Amplitude Change (AAC)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2


Difference Absolute Standard Deviation Value (DASDV)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2


Modified Mean Absolute Value (MMAV)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2


Modified Mean Absolute Value 3 (MMAV2)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2


Variance of EMG (VAR)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2

12.3 Anexo 3. Resultados AUC combinación características

n	EMAV	MAV	RST	AAC	DASDV	MMAV	MMAV2	VAR	AUC	Desviación
2	1	1							0,75407	0,024302
2	1		1						0,75003	0,032708
2	1			1					0,8075	0,021733
2	1				1				0,80656	0,02249
2	1					1			0,76691	0,023714
2	1						1		0,75861	0,030052
2	1							1	0,76531	0,024558
2		1	1						0,79806	0,0329
2		1		1					0,80616	0,021859
2		1			1				0,80511	0,022578
2		1				1			0,77475	0,027552
2		1					1		0,76586	0,029017
2		1						1	0,76637	0,028468
2			1	1					0,81107	0,022765
2			1		1				0,81001	0,022787
2			1			1			0,79337	0,033898
2			1				1		0,78425	0,029524
2			1					1	0,78582	0,032165
2				1	1				0,80235	0,023257
2				1		1			0,80673	0,021855
2				1			1		0,81325	0,0251
2				1				1	0,81524	0,022482
2					1	1			0,80685	0,022095
2					1		1		0,81311	0,025714
2					1			1	0,81717	0,022845
2						1	1		0,76146	0,032017
2						1		1	0,75453	0,031825
2							1	1	0,76693	0,024922
3	1	1	1						0,78811	0,028536
3	1	1		1					0,78835	0,029766
3	1	1			1				0,79072	0,030744
3	1	1				1			0,7702	0,024617
3	1	1					1		0,754	0,023936
3	1	1						1	0,75611	0,024526
3	1		1	1					0,80016	0,027543
3	1		1		1				0,79899	0,028955
3	1		1			1			0,79132	0,027607
3	1		1				1		0,76923	0,033221
3	1		1					1	0,79168	0,036006
3	1			1	1				0,79779	0,024475
3	1			1		1			0,79419	0,031554
3	1			1			1		0,80348	0,02362
3	1			1				1	0,79719	0,029877
3	1				1	1			0,79612	0,031548

3	1				1		1		0,80338	0,024678
3	1				1			1	0,79823	0,030773
3	1					1	1		0,77595	0,020181
3	1					1		1	0,76766	0,023359
3	1						1	1	0,76431	0,025033
3		1	1	1					0,82692	0,022733
3		1	1		1				0,8272	0,02257
3		1	1			1			0,79588	0,034084
3		1	1				1		0,79555	0,029561
3		1	1					1	0,79356	0,035144
3		1		1	1				0,80546	0,025445
3		1		1		1			0,81274	0,020937
3		1		1			1		0,81251	0,025275
3		1		1				1	0,80073	0,029576
3		1			1	1			0,81266	0,021328
3		1			1		1		0,81293	0,026301
3		1			1			1	0,80015	0,030434
3		1				1	1		0,77134	0,029266
3		1				1		1	0,77657	0,033685
3		1					1	1	0,76593	0,028569
3			1	1	1				0,80952	0,026342
3			1	1		1			0,82441	0,019032
3			1	1			1		0,81089	0,021491
3			1	1				1	0,80902	0,028887
3			1		1	1			0,82492	0,018726
3			1		1		1		0,80973	0,022241
3			1		1			1	0,8096	0,030009
3			1			1	1		0,79255	0,032348
3			1			1		1	0,79456	0,033156
3			1				1	1	0,78654	0,032648
3				1	1	1			0,80857	0,023588
3				1	1		1		0,81389	0,024274
3				1	1			1	0,81648	0,022863
3				1		1	1		0,82593	0,026416
3				1		1		1	0,79674	0,027229
3				1			1	1	0,81296	0,022409
3					1	1	1		0,82734	0,02633
3					1	1		1	0,79604	0,028617
3					1		1	1	0,81151	0,024477
3						1	1	1	0,75677	0,029198
4	1	1	1	1					0,81528	0,02928
4	1	1	1		1				0,81626	0,028908
4	1	1	1			1			0,78836	0,028798
4	1	1	1				1		0,78813	0,028593
4	1	1	1					1	0,7883	0,028605
4	1	1		1	1				0,78915	0,035058
4	1	1		1		1			0,79609	0,030501

4	1	1		1			1		0,80127	0,024213
4	1	1		1				1	0,78342	0,029556
4	1	1			1	1			0,79844	0,030215
4	1	1			1		1		0,80423	0,024721
4	1	1			1			1	0,78613	0,030539
4	1	1				1	1		0,7784	0,02082
4	1	1				1		1	0,77002	0,02452
4	1	1					1	1	0,75581	0,02441
4	1		1	1	1				0,79004	0,032967
4	1		1	1		1			0,81352	0,024553
4	1		1	1			1		0,80052	0,025585
4	1		1	1				1	0,81449	0,02331
4	1		1		1	1			0,81361	0,024322
4	1		1		1		1		0,7998	0,027249
4	1		1		1			1	0,81463	0,023845
4	1		1			1	1		0,79211	0,022788
4	1		1			1		1	0,79299	0,032429
4	1		1				1	1	0,79125	0,034628
4	1			1	1	1			0,79296	0,030946
4	1			1	1		1		0,79642	0,025532
4	1			1	1			1	0,79481	0,0314
4	1			1		1	1		0,82746	0,022142
4	1			1		1		1	0,79337	0,029341
4	1			1			1	1	0,80432	0,025649
4	1				1	1	1		0,82905	0,021621
4	1				1	1		1	0,79558	0,029161
4	1				1		1	1	0,80472	0,026419
4	1					1	1	1	0,77608	0,020268
4		1	1	1	1				0,82618	0,022069
4		1	1	1		1			0,82654	0,020394
4		1	1	1			1		0,82935	0,022473
4		1	1	1				1	0,81863	0,023436
4		1	1		1	1			0,82697	0,020248
4		1	1		1		1		0,83073	0,02203
4		1	1		1			1	0,81852	0,023473
4		1	1			1	1		0,79517	0,029649
4		1	1			1		1	0,79433	0,034038
4		1	1				1	1	0,79303	0,034495
4		1		1	1	1			0,80894	0,024455
4		1		1	1		1		0,8129	0,024587
4		1		1	1			1	0,79541	0,027757
4		1		1		1	1		0,8401	0,02451
4		1		1		1		1	0,80864	0,026199
4		1		1			1	1	0,80816	0,024371
4		1			1	1	1		0,84078	0,024144
4		1			1	1		1	0,80923	0,026833
4		1			1		1	1	0,80729	0,025142

4		1				1	1	1	0,78162	0,025565
4			1	1	1	1			0,82366	0,02171
4			1	1	1		1		0,80847	0,025739
4			1	1	1			1	0,80781	0,028194
4			1	1		1	1		0,83327	0,022013
4			1	1		1		1	0,8162	0,024719
4			1	1			1	1	0,80777	0,029182
4			1		1	1	1		0,83474	0,02169
4			1		1	1		1	0,81624	0,024917
4			1		1		1	1	0,80745	0,029888
4			1			1	1	1	0,79469	0,031378
4				1	1	1	1		0,82682	0,024078
4				1	1	1		1	0,788	0,032173
4				1	1		1	1	0,80231	0,031066
4				1		1	1	1	0,81711	0,024364
4					1	1	1	1	0,8167	0,02617
5	1	1	1	1	1				0,81225	0,032231
5	1	1	1	1		1			0,81512	0,027996
5	1	1	1	1			1		0,82165	0,025986
5	1	1	1	1				1	0,81386	0,028749
5	1	1	1		1	1			0,81631	0,027839
5	1	1	1		1		1		0,82306	0,025969
5	1	1	1		1			1	0,81453	0,028756
5	1	1	1			1	1		0,79007	0,02662
5	1	1	1			1		1	0,78883	0,028569
5	1	1	1				1	1	0,78834	0,028462
5	1	1		1	1	1			0,79388	0,029679
5	1	1		1	1		1		0,80394	0,02915
5	1	1		1	1			1	0,78445	0,033964
5	1	1		1		1	1		0,82943	0,022622
5	1	1		1		1		1	0,7896	0,029334
5	1	1		1			1	1	0,79801	0,024233
5	1	1			1	1	1		0,83063	0,022436
5	1	1			1	1		1	0,7913	0,029023
5	1	1			1		1	1	0,80116	0,024598
5	1	1				1	1	1	0,7787	0,020817
5	1		1	1	1	1			0,80534	0,025516
5	1		1	1	1		1		0,79155	0,03109
5	1		1	1	1			1	0,81052	0,024172
5	1		1	1		1	1		0,8288	0,022805
5	1		1	1		1		1	0,81486	0,026023
5	1		1	1			1	1	0,81063	0,024238
5	1		1		1	1	1		0,82966	0,023532
5	1		1		1	1		1	0,81508	0,025937
5	1		1		1		1	1	0,80962	0,024387
5	1		1			1	1	1	0,79485	0,026789
5	1			1	1	1	1		0,82552	0,023869

5	1			1	1	1		1	0,79239	0,02888
5	1			1	1		1	1	0,79995	0,029874
5	1			1		1	1	1	0,82594	0,021898
5	1				1	1	1	1	0,82779	0,021881
5		1	1	1	1	1			0,8251	0,02175
5		1	1	1	1		1		0,83106	0,021095
5		1	1	1	1			1	0,81414	0,024124
5		1	1	1		1	1		0,83592	0,022842
5		1	1	1		1		1	0,81756	0,024232
5		1	1	1			1	1	0,81579	0,026257
5		1	1		1	1	1		0,83709	0,022741
5		1	1		1	1		1	0,81729	0,024209
5		1	1		1		1	1	0,81598	0,026098
5		1	1			1	1	1	0,79485	0,032152
5		1		1	1	1	1		0,83857	0,027555
5		1		1	1	1		1	0,80357	0,024866
5		1		1	1		1	1	0,79931	0,029749
5		1		1		1	1	1	0,82414	0,030703
5		1			1	1	1	1	0,8236	0,0319
5			1	1	1	1	1		0,83664	0,022929
5			1	1	1	1		1	0,81241	0,025303
5			1	1	1		1	1	0,80475	0,028531
5			1	1		1	1	1	0,82172	0,029312
5			1		1	1	1	1	0,82127	0,029496
5				1	1	1	1	1	0,80875	0,030855
6	1	1	1	1	1	1			0,81122	0,030795
6	1	1	1	1	1		1		0,81765	0,032322
6	1	1	1	1	1			1	0,81075	0,032804
6	1	1	1	1		1	1		0,82818	0,023425
6	1	1	1	1		1		1	0,8131	0,028095
6	1	1	1	1			1	1	0,82086	0,025422
6	1	1	1		1	1	1		0,82891	0,023686
6	1	1	1		1	1		1	0,81388	0,028293
6	1	1	1		1		1	1	0,8224	0,025841
6	1	1	1			1	1	1	0,79019	0,026296
6	1	1		1	1	1	1		0,82679	0,025991
6	1	1		1	1	1		1	0,78856	0,028162
6	1	1		1	1		1	1	0,80105	0,02884
6	1	1		1		1	1	1	0,82798	0,022541
6	1	1			1	1	1	1	0,82929	0,022282
6	1		1	1	1	1	1		0,82567	0,026917
6	1		1	1	1	1		1	0,80842	0,026293
6	1		1	1	1		1	1	0,80622	0,025458
6	1		1	1		1	1	1	0,82817	0,023574
6	1		1		1	1	1	1	0,82891	0,024261
6	1			1	1	1	1	1	0,82574	0,023845
6		1	1	1	1	1	1		0,83779	0,026297

6		1	1	1	1	1		1	0,81289	0,024815
6		1	1	1	1		1	1	0,81351	0,027361
6		1	1	1		1	1	1	0,8214	0,030671
6		1	1		1	1	1	1	0,82095	0,031815
6		1		1	1	1	1	1	0,82004	0,0339
6			1	1	1	1	1	1	0,81891	0,031615
7	1	1	1	1	1	1	1		0,8248	0,027273
7	1	1	1	1	1	1		1	0,80947	0,031549
7	1	1	1	1	1		1	1	0,81747	0,032472
7	1	1	1	1		1	1	1	0,82762	0,023279
7	1	1	1		1	1	1	1	0,82857	0,023892
7	1	1		1	1	1	1	1	0,82626	0,025131
7	1		1	1	1	1	1	1	0,82514	0,026732
7		1	1	1	1	1	1	1	0,81898	0,034185
8	1	1	1	1	1	1	1	1	0,82426	0,027343