industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

Universidad Politécnica de Cartagena

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

INTEGRACIÓN DEL DISPOSITIVO EMOTIV EPOC EN UNA APLICACIÓN BRAIN COMPUTER INTERFACE ASOCIADA AL CONTROL DE UN EXOESQUELETO

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Pablo Pindado Herráez**
Directores: Dr. José Manuel Cano Izquierdo
Dr. Julio José Ibarrola Lacalle

Cartagena, 6 de abril de 2021

Resumen

El presente Trabajo Fin de Grado forma parte de un proyecto más amplio, cuyo principal objetivo la implementación de un Sistema BCI (*Brain-Computer Interface* o Interfaz Cerebro-Ordenador) para la utilización de un robot tipo exoesqueleto que asista a personas con movilidad reducida u otras discapacidades motoras. Para lograr el objetivo, abordaremos distintas áreas del conocimiento, como son la electroencefalografía, las tecnologías BCI y el entorno de cálculo numérico MATLAB, así como un desarrollo inicial de los fundamentos teóricos que permiten funcionar a los sistemas utilizados para que el lector pueda familiarizarse con la terminología de los mismos.

Palabras clave: Exoesqueleto, Tren inferior, Sistema BCI, Emotiv EPOC, Electroencefalografía, MATLAB

Abstract

This Final Undergraduate Project belonks to a bigger proyect which seeks to implement a BCI System (Brain-Computer Interface) to control a exoskeleton robot that can help disabled people and people with reduced mobility. To reach this goal, we will expose a study in different areas and fields of knowledge, such as electroencephalography (EEG onwards), BCI technologies and MATLAB software, as well as a theoretical development about the fundaments and operation of the systems and elements used in this Project.

Keywords: Exoskeleton, Lower limb, BCI System, Emotiv EPOC, Electroencephalograph, MATLAB

Índice de contenido

Índice de Ilustraciones

Índice de tablas

1. Introducción y objetivos del Proyecto

Una de las líneas de investigación en el entorno de la ciencia y la medicina se relaciona con el estudio de las personas con discapacidades motoras, ya sean por malformaciones genéticas o por lesiones medulares irreversibles. Sólo en España existen alrededor de 35.000 personas con lesiones medulares que impiden desarrollar funciones motoras, ya sean del tren inferior o de ambos trenes superior e inferior. Además, se estima que cada año hay entre 8000 y 12000 personas más con dicha discapacidad [1].

Una persona con tetraplejia ve cómo su vida depende casi completamente del cuidador al cargo de la misma, por lo que su calidad de vida se ve reducida considerablemente una vez se presenta dicha discapacidad [2]. Los estudios más recientes revelan un creciente interés por los sistemas de tipo BCI (*brain-computer interface,* o interfaz cerebro-ordenador) como una solución a corto plazo para paliar dichas dependencias de personas con lesiones medulares respecto de sus cuidadores, implementando controles BCI sobre sillas de ruedas [3] [4] [5] [6] [7].

Por otro lado, la electroencefalografía ha seguido una tendencia de uso creciente en la sociedad, pasando de ser una tecnología reservada a la investigación en campos de la psicología, medicina y la neurociencia con equipos complejos, con un elevado coste y de fabricación limitada, a encontrar en el mercado equipos EEG de bajo coste y fácil uso y colocación, que integran interfaces sencillas y didácticas adaptadas a un usuario medio que ve así reducida la curva de aprendizaje necesaria para usar este tipo de tecnología. Es por ello que este tipo de equipos de bajo coste han despertado el interés de investigadores acerca de la posibilidad de implementación de los mismos como equipo de adquisición de datos en sistemas que requieran una elevada fiabilidad, como un sistema BCI para el control de un exoesqueleto.

Ilustración 1. Exoesqueleto EXO-LEGS, disponible en el Dpto. Automática, Ingeniería Eléctrica y Tecnología Electrónica de la UPCT [8]

1.1. Motivaciones del Proyecto

Este Proyecto nace bajo la idea de establecer una conexión entre dos líneas de investigación recientes en el Departamento de Ingeniería Automática de la Universidad Politécnica de Cartagena. Por un lado, nos encontramos los Proyectos de investigación relacionados con la implementación de equipos de Electroencefalografía de bajo coste para la predicción de pensamiento del usuario. Por otro lado, existe un Proyecto de adecuación y puesta en funcionamiento del exoesqueleto EXO-LEGS, desarrollado bajo el programa europeo Ambient Assisted Living Joint Programme (AAL-010000-2012-15) (Véase Ilustración 1). Ambos campos están siendo abordados por profesorado y alumnado a través de distintos Trabajos Fin de Estudio y Becas de Colaboración [9] [10] [11] [12] [13].

1.2. Objetivos del Proyecto

Una vez fijadas las ideas y motivaciones que generan el planteamiento de este Proyecto, podemos prefijar un conjunto de objetivos a cumplir para que se considere el mismo como finalizado o logrado. Estos objetivos son los siguientes:

- Obtener un *software* que permita la adquisición y registro de datos EEG en relación a la intención de movimiento de un usuario a partir del equipo Emotiv EPOC+.
- Desarrollar un sistema a partir de modelos de predicción que permita, a través de los datos registrados, inferir estados del pensamiento del usuario acerca de la intención de caminar.
- Implementar un sistema que emplee dichos datos para controlar un exoesqueleto real de tren inferior en tiempo real.
- Lograr un porcentaje de acierto del sistema aceptable en relación a la calidad del equipo y del software empleado.

Para alcanzar dichos objetivos, será necesario abordar diversas técnicas del mundo de la Neurociencia, la Electrónica y Automática, Control por Computador y Robótica., como se verá en los siguientes epígrafes.

2. Antecedentes

A lo largo del siguiente epígrafe, se desarrollarán las bases científicas y tecnológicas necesarias para comprender el trabajo realizado. Dado que éste abarca distintas áreas de la ciencia, como es la neurociencia, la electrónica, la informática y el control por computador, se hace necesario recalcar la teoría básica de la misma en los siguientes epígrafes.

2.1. El cerebro humano

La base de este Proyecto es la detección de patrones realizados durante la actividad cerebral de un usuario a través de un equipo como es el Emotiv EPOC. El cerebro es un órgano que pertenece al sistema nervioso del cuerpo humano, siendo el responsable de recibir todos los impulsos de los sentidos a través de la médula espinal, de enviar respuestas a dichos impulsos, además de la gestión de emociones, imaginación, memoria e inteligencia. Dichi órgano posee diferentes zonas dedicadas a tareas específicas, como se puede observar en la Ilustración 2.

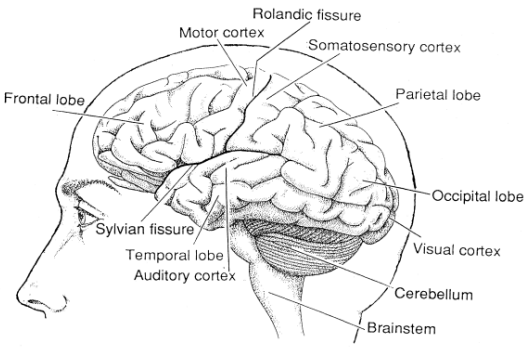

Ilustración 2. Esquema del cerebro humano visto desde el lado izquierdo, junto con las zonas estructurales más importantes y áreas especiales del cortex cerebral [14]

Estas tareas se realizan gracias al funcionamiento coordinado de alrededor de 100.000.000.000 neuronas, unas células que son capaces de transmitir información a gran velocidad a través de ellas gracias al uso de señales eléctricas, permitiendo así que se comuniquen unas con otras en un mecanismo que se conoce como sinapsis [15]. Una célula presenta la siguiente estructura: (Véase la Ilustración 3)

- Núcleo
- Soma o pericarion rodeando al núcleo
- Dendritas
- Axón, encargado de conducir el impulso nervioso desde el pericarion a otra neurona

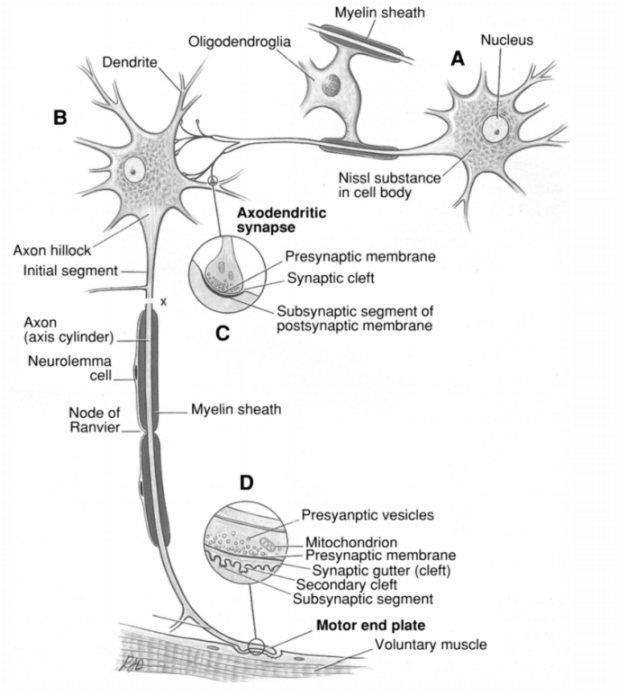

Ilustración 3. Estructura básica de una neurona [15]

Estas neuronas pueden tener una sustancia rodeando al axón llamada mielina, encargada de aumentar la resistencia de la membrana axónica y, por tanto, aumenta también al velocidad de transmisión a través del mismo. La presencia o no de esta sustancia en el axón de una neurona genera dos colores diferenciales en un cerebro: Las zonas conocidas como *materia gris* y *materia blanca.* (Véase la Ilustración 4)

- La materia gris está compuesta principalmente por axones carentes de mielina. Estas zonas se asocian a las tareas de procesado de información o razonamiento debido a la baja velocidad de transmisión.
- La materia blanca, al contrario de la materia gris, está formada por fibras nerviosas cubiertas mielinizadas. Debido a la elevada velocidad de transmisión que puede poseer una neurona mielinizada, estas zonas cerebrales se asocian con actividades de retransmisión, coordinación y comunicación entre diferentes regiones del cerebro [16].

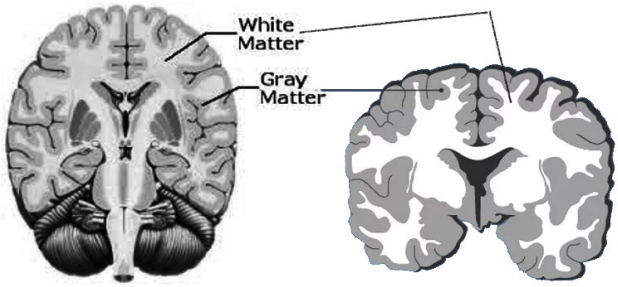

Ilustración 4. Materia gris y materia blanca en un corte del cerebro humano

A su vez, el cerebro se puede dividir, según su posición y funcionalidades, en cuatro regiones o lóbulos: (Véase la Ilustración 5)

- Lóbulo frontal, encargado de la conducta humana (aprendizaje, pensamiento, razonamiento, pensamiento, emociones, articulación del lenguaje y juicios).
- Lóbulo occipital, encargado de las señales del sentido de la vista y de la transmisión de las mismas al resto de lóbulos.
- Lóbulo parietal, encargado de procesar la información sensorial que llega de todas las partes del cuerpo y del control del movimiento.
- Lóbulo temporal, relacionado con la memoria y el reconocimiento de ciertos patrones sensoriales como olores, voces y otras sensaciones captadas por los sentidos.

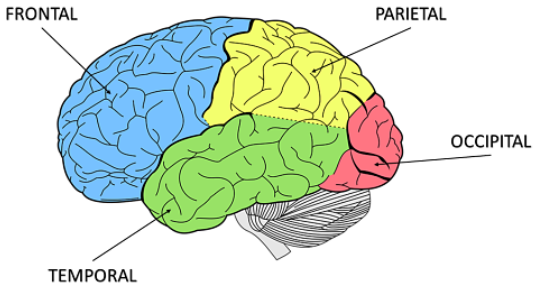

Ilustración 5. Lóbulos de un cerebro humano

2.2. Procedimientos de medida de actividad cerebral

Hemos visto en el Apartado 2.1. que el elemento básico de dicho órgano es la neurona, una célula que transmite información a través de pulsos eléctricos para comunicarse con las que le rodean. Las ondas cerebrales son las actividades eléctricas que produce el cerebro como

resultado de la transmisión de información a través de los axones de las neuronas. Se tratan de ondas de tensión de baja amplitud ($\sim 10 - 100 \mu V$) y frecuencias dentro del rango de 1 a 100 Hz:

- Ondas *delta* (1-3 Hz), relacionadas con el tiempo de sueño.
- Ondas *theta* (3.1-7.9 Hz), asociadas con las primeras etapas del sueño.
- Ondas *alpha* (8-13 Hz), ocasionadas por las actividades síncronas y coherentes
- Ondas *beta* (14-29 Hz), relacionadas con etapas de sueño nulo, períodos previos al despertar y también asociadas con el comportamiento motor, por ejemplo, con el inicio y final de un movimiento voluntario [17].
- Ondas *gamma* (30-100 Hz)

Dichas señales pueden ser capturadas y registradas para analizar posteriormente si una región participa en el control de diversas conductas, como pueden ser presentaciones de estímulos, tomas de decisiones o ejecución de actividades motoras [18]. Existen, según la extensión de la medición, tres tipos de registros: (Véase la Ilustración 6)

- Registro de neuronas individuales mediante microelectrodos: Un microelectrodo es un sensor con una punta muy fina que es capaz de registrar la actividad de una neurona o un grupo reducido de neuronas de manera individual. Estos electrodos son implantados mediante cirugía en el encéfalo y registran unas señales muy débiles que deben ser posteriormente amplificadas.
- Registro con macroelectrodos: Un macroelectrodo, al contrario que un microelectrodo, no detecta la actividad de neuronas individuales, sino que sus lecturas representan los potenciales postsinápticos de miles o millones de células del área en torno a dicho electrodo. Estos electrodos pueden consistir en alambres insertados en el encéfalo mediante cirugía, tornillos fijados en el cráneo o discos situados sobre el cuero cabelludo acompañados de un componente que reduce la impedancia presente en el contacto del mismo. Esta actividad eléctrica registrada puede ser representada temporalmente, lo que se conoce como electroencefalografía (EEG)
- Registro con neuromagnetómetros: La magnetoencefalografía es la técnica que se aprovecha de los campos magnéticos que inducen las corrientes eléctricas que fluyen a través de los axones. A partir de unos detectores superconductores llamados SQUID (dispositivos superconductores de interferencias cuánticas)

podemos detectar campos magnéticos inapreciables para otros sensores magnéticos. La principal ventaja de la magnetoencefalografía respecto a otras técnicas de adquisición de datos cerebrales es la elevada resolución temporal que presentan [14].

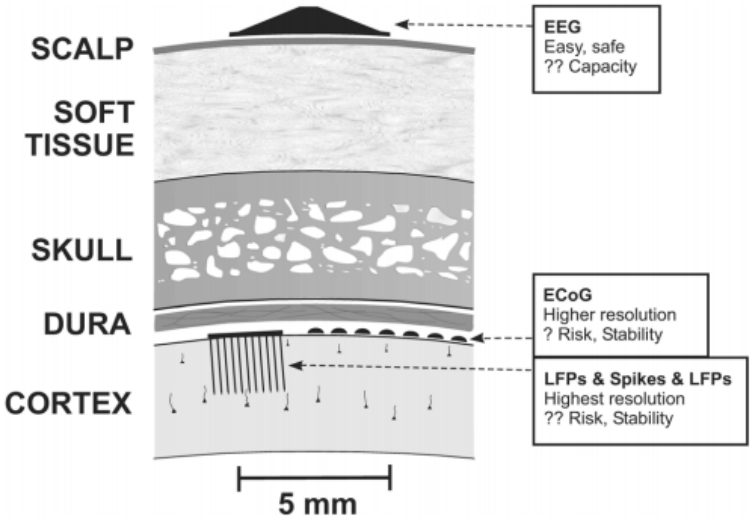

Ilustración 6. Posición de sensores EEG (sobre el cuero cabelludo), ECoG (superficie del cortex) y LFP (dentro del cerebro) [19]

2.3. La electroencefalografía (EEG)

La electroencefalografía es una técnica de medición de la actividad cerebral de tipo no invasiva (es decir, que no requiere de cirugía para implantar los sensores). Su fundamento teórico es la grabación y monitorización de diferencias de potenciales eléctricos entre distintos puntos de la superficie del cuero cabelludo, derivados de los potenciales postsinápticos excitadores e inhibidores que generan las neuronas, como pudimos ver en el Apartado 2.1. El cerebro humano [20].

Se trata de una técnica que emplea macroelectrodos (véase el Apartado 2.2. Procedimientos de medida de actividad cerebral), por lo que la medición obtenida no será de un grupo reducido de neuronas, sino de grandes zonas neuronales del cerebro. La electroencefalografía es ampliamente usada en distintos campos de la neurociencia, con diversas aplicaciones clínicas como el estudio de ataques epilépticos frente a otros episodios similares como migrañas o síncopes [21], o entre encefalopatías y estados psíquicos [22].

2.3.1. Posiciones de los electrodos

Como hemos comentado anteriormente, un EEG posee uno o varios macroelectrodos. Dichos sensores suelen colocarse en la superficie del cráneo siguiendo un patrón normalizado

según el *Sistema 10-20,* el cual recibe el nombre debido a que las distancias reales entre los electrodos adyacentes son un 10% y 20% de la distancia total de adelante hacia atrás o de derecha a izquierda, respectivamente. Este método se desarrolló con el objetivo de poder realizar ensayos de EEG universales y estandarizados para que los resultados puedan ser comparados, reproducidos y analizados usando el método científico [23].

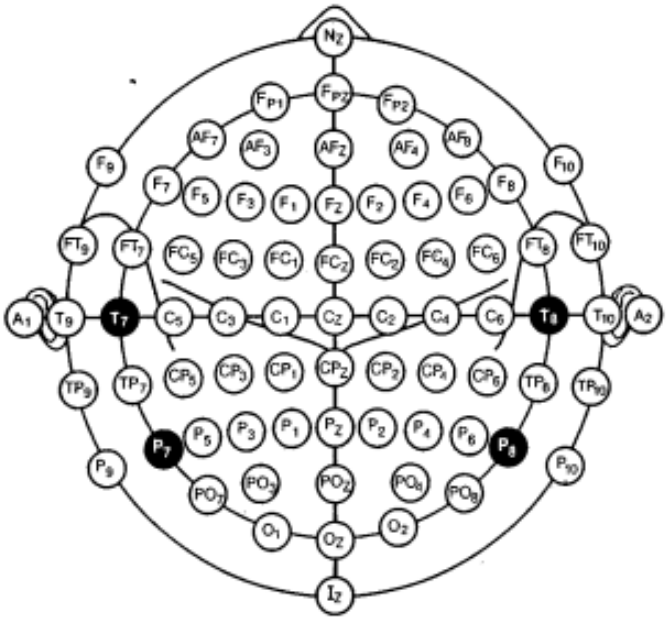

Ilustración 7. Posiciones de los electrodos según el sistema internacional 10-20 para grabaciones de ensayos mediante EEG [23]

Como podemos apreciar en la Ilustración 7, cada posición se codifica de acuerdo a las siguientes normas:

1. Cada posición recibe, de inicio, un total de una o dos letras, según el sensor corresponda a un lóbulo o dos (Véase el Apartado 2.1. El cerebro humano):
    a. Lóbulo frontal: Letra F
    b. Lóbulo occipital: Letra O
    c. Lóbulo parietal: Letra P
    d. Lóbulo temporal: Letra T
2. Cada posición recibe, después de la determinación del lóbulo o lóbulos correspondiente, un número o letra ‘z’ según su posición sagital respecto al centro. De esta manera, los sensores ubicados en el eje de simetría sagital reciben una ‘z’ de ‘zero’, los que se sitúan a la izquierda del eje de simetría reciben dígitos impares y los que se sitúan a la derecha del eje de simetría, valores pares.

Existen otras nomenclaturas para las posiciones de los electrodos muy conocidas, como puede ser el *Sistema 10-5*.

2.3.2. Tratamiento de señales EEG

Existe un problema evidente [24] a la hora del análisis de datos EEG brutos, y es que tan solo el 50% de la contribución de los datos procede de fuentes en un radio de tres centímetros respecto a dicho electrodo. Esto es debido a distintos factores:

1. Artefactos: Potenciales que se presentan en las lecturas de los datos EEG que proceden de señales eléctricas de origen no cerebral, como pueden ser parpadeos del ojo, artefactos musculares o latidos del corazón.
2. Ruido electrónico: Señales eléctricas residuales inducidas por aparatos electrónicos en las frecuencias coincidentes a las señales cerebrales. Los ruidos más comunes son los inducidos por la red eléctrica. En el caso de España, la red eléctrica opera a una frecuencia de 50Hz, por lo que afectaría a la captación de ondas gamma.

Para ello, se suelen emplear distintos filtros para conseguir limpiar dichos datos en bruto de los ruidos e impurezas que se hayan podido infiltrar dentro de la señal. Los más conocidos son los siguientes:

1. Filtro Laplaciano: También conocido como filtro Gaussiano, es un filtro ampliamente utilizado en el procesado de imagen. La ecuación de un filtro Laplaciano sería la siguiente:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

   Dicha ecuación representa que el valor que adquiera un sensor en un instante va a verse influido también por los valores que adquieran sus sensores más próximos.
2. *Common Average Reference (CAR)*: CAR es una técnica de filtrado que asocia el potencial de cada electrodo respecto a la medida de todos los electrodos en cada instante. La expresión de un filtro CAR sería la siguiente: [25]

$$V_i^{CAR} = V_i^{ER} - \frac{1}{n}\sum_{j=1}^{n} V_j^{ER}$$

3. *Common Spatial Pattern (CSP)*: Un filtro CSP es un método que se encarga de separar una señal multivariable en subcomponentes aditivas que posean una varianza máxima entre dos ventanas. De esta manera, se consigue separar una señal en bruto en distintas señales con orígenes distintos.

2.3.3. Potenciales evocados

Un potencial evocado (PE) es una fluctuación en el voltaje de un EEG provocada por un suceso sensorial, motor o cognitivo. Estos PE suelen adoptar formas de picos o valles, como se puede observar en un PE estudiado en la Ilustración 8, y nos indican que ha ocurrido dicho proceso cognitivo [26].

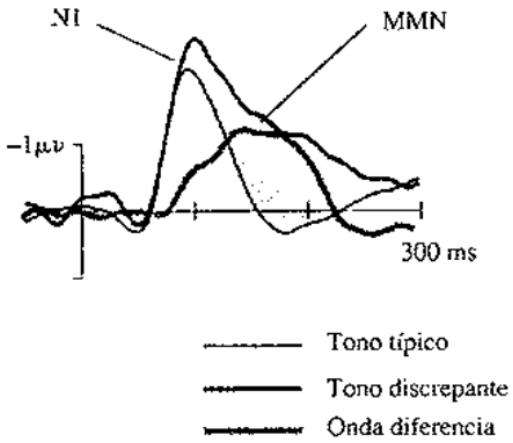

Ilustración 8. PE ante un tono auditivo típico y un tono discrepante de baja probabilidad [26]

Según el tipo de estímulo que genera el potencial, los PE suelen clasificarse en:

1. PE Visuales (VEP): Cambios en el potencial debidos a estimulaciones visuales.
2. PE Auditivos (AEP): Cambios en el potencial generados a partir de estímulos sonoros.
3. PE Somato sensoriales: Cambios en el potencial provocados a partir de estímulos eléctricos.

En la electroencefalografía, un potencial eléctrico muy importante es el conocido P300, un pico de polaridad positiva que aparece ante una gran variedad de estímulos. La letra 'P' hace referencia a que se trata de un PE positivo, mientras que el 300 define la latencia del proceso cognitivo en cuestión, en milisegundos, aunque también ha llegado a presentarse en latencias de hasta 700 ms. En la Ilustración 9, podemos observar cómo afecta la amplitud de un estímulo P300 en cuanto a la probabilidad de que ese PE se deba a un estímulo en concreto.

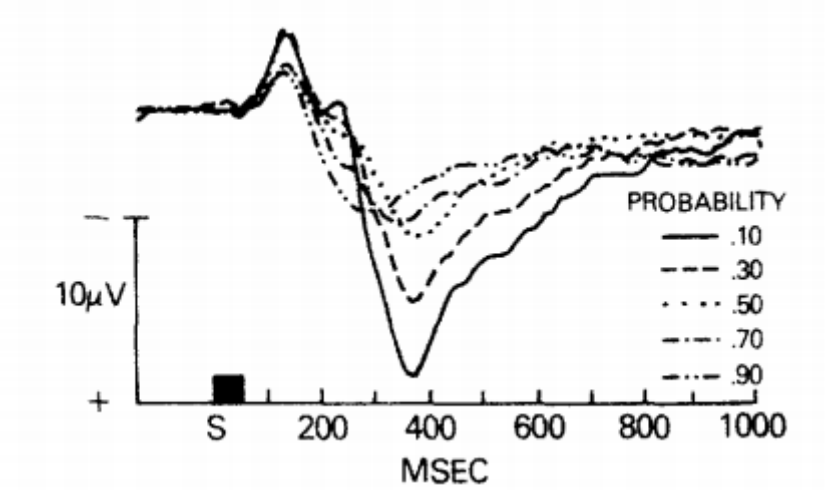

Ilustración 9. Relación entre la amplitud del estímulo P300 y la probabilidad de que se deba a un estímulo [27]

2.4. Interfaces Cerebro-Ordenador (BCI)

Una interfaz cerebro-ordenador, también conocida como interfaz neuronal directa, o más comúnmente por sus siglas en inglés BCI (*Brain-Computer interface*) es un dispositivo que permite establecer una conexión entre el mundo externo a partir de la actividad eléctrica que se puede registrar en el cerebro, permitiendo así evitar el uso de actividades motoras como pueden ser pulsaciones de botones, accionamiento de *joysticks*, desplazamiento de controles como un ratón de ordenador, etc. (Véase la Ilustración 10). Estos sistemas son especialmente interesantes cuando se emplean en casos de personas con discapacidades motoras severas como las comentadas en la Introducción de la literatura [28].

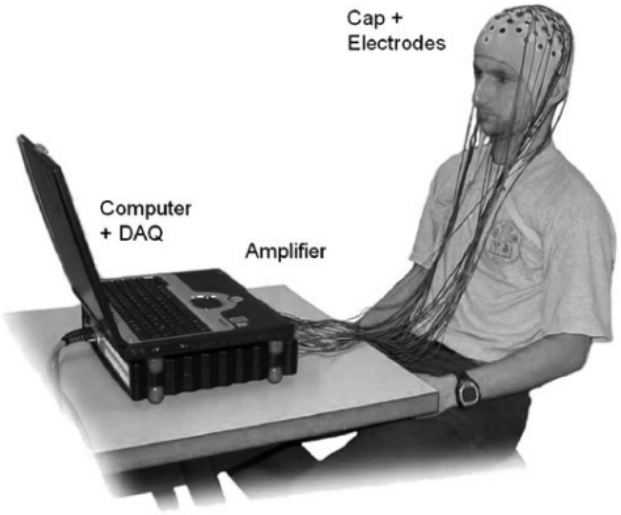

Ilustración 10. Ejemplo de equipo BCI basado en EEG, consistente en un ordenador con una tarjeta de adquisición de datos, un amplificador para las señales EEG y un casco cerebral de tipo malla con electrodos repartidos por la superficie del cuero cabelludo [29]

Un BCI debe estar compuesto por cuatro componentes básicos [29]:

- Debe grabar la actividad cerebral directamente desde el cerebro, de manera invasiva o no invasiva (relativo a la necesidad o no de intervención quirúrgica para la colocación del electrodo, respectivamente).
- Debe existir una realimentación o *feedback* con el usuario.
- El *feedback* debe ser en tiempo real.
- El usuario debe tener la capacidad de realizar una tarea mental de forma intencional, elegida por él mismo, que suponga una meta para el BCI

2.4.1. Funcionamiento de un BCI

El mecanismo básico de un BCI es la medición de la actividad cerebral, junto al posterior procesado de la misma para producir señales de control asociadas con las intenciones del usuario. La mayoría de los BCI se apoyan en las señales EEG debido a su facilidad de aplicación, bajo coste y ligereza frente a otros tipos de tecnologías también usadas como pueden ser la electrocorticografía (ECoG), la cual necesita cirugía. Una vez grabada la señal EEG, se analiza frente a patrones cerebrales para intentar obtener conclusiones por semejanza [29].

Para que un BCI funcione correctamente, es fundamental aplicar ciertas estrategias mentales por parte del usuario, con el fin de generar patrones diferenciales que sean fácilmente desencriptables por el BCI. Las técnicas más empleadas son:

1. Atención selectiva: Uso de estímulos externos de tipo auditivo, somatosensorial o visual, con el fin de generar un estado de concentración máxima en el usuario hacia dicho estímulo. Estos estímulos están relacionados con el potencial P300 comentado en el Apartado 2.3.3.
2. Imaginación motora: Cuando una persona comienza a preparar el movimiento de un brazo, de una pierna o cualquier otra extremidad, se generan cambios en los ritmos sensorimotores (SMR). Dichos ritmos, cuando se extraen de las oscilaciones *alpha* (véase el Apartado 2.2. Procedimientos de medida de actividad cerebral), reciben el nombre de actividad *mu*. Estas señales *mu*, junto a las señales *beta*, son las más importantes en lo relativo a imaginación motora en EEG. Sin embargo, la imaginación motora no es tan efectiva a corto plazo como la atención selectiva, puesto que para que funcionen correctamente debe ser precedido por un entrenamiento por parte del usuario.

3. Adquisición de datos del equipo Emotiv EPOC

El Proyecto desarrollado a lo largo de toda la literatura emplea como hardware de adquisición de datos EEG un Emotiv EPOC+ v1, de 2008, de venta al público, mostrado en la Ilustración 11. Una de las ventajas de la implementación de este *headset* es que Emotiv, como fabricante del equipo EPOC, proporciona un kit de desarrollo de software (*software development kit* o SDK) que nos permite conectarnos al equipo y obtener los datos que los sensores del mismo están tomando en cada instante. Este equipo consta de 14 electrodos de tipo húmedo, situados sobre las posiciones AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4 según establece el sistema estandarizado internacional 10-20, además de contar con dos almohadillas de referencia situadas en las posiciones P3 y P4. Gracias al trabajo desarrollado por el Dr. Juan Antonio Martínez León en su Tesis Doctoral [30] disponemos de scripts desarrollados para el software 'Matlab' que nos permiten realizar la toma y almacenado de los datos proporcionados por el equipo.

Ilustración 11. Emotiv EPOC+ v1, equipo EEG usado para el desarrollo del Proyecto

Estos scripts están desarrollados bajo el marco de la 'BCI Competition II', la cual marca unos requisitos de funcionamiento muy restrictivos, además de fijar como objetivo el movimiento diferencial de cada brazo y el pensamiento de una palabra aleatoria. Es por ello que deberemos adaptar este software a nuestras especificaciones personales para intención de caminar. El bloque de scripts cuenta con tres grupos principales:

- La adquisición de datos, que será el subconjunto de scripts que nos permitirán obtener los datos en bruto (*raw data*) en cuatro grupos de datos para los cuatro

experimentos a realizar por cada persona: ‘Exp1.csv’, ‘Exp2.csv’, ‘Exp3.csv’ y ‘Exp4.csv’. Estas tareas son realizadas ejecutando el algoritmo ‘EEGRecordingPanel1.m’. Los datos son los potenciales eléctricos que genera la actividad cerebral, comprendidos en un rango entre 10 y 100 μV.

- El preprocesado matricial, que, a partir de los datos ‘raw’ obtenidos en la adquisición de datos, será el subconjunto de scripts encargado de realizar un filtrado de los potenciales adquiridos a partir de una superficie Laplaciana para eliminar interferencias causadas por ruidos y medidas no deseadas, y un posterior cálculo de la Densidad Espectral de Potencia (PSD) entre las bandas 8-30 Hz con saltos de 2 Hz (12 frecuencias en total) en ventanas de un segundo desde la medición actual.
- Procesamiento de los datos basado en un sistema de control tipo *Supervised dFasArt* (*Supervised and Dynamic Fuzzy Adaptative System ART-base*) para, a partir de la combinación de modelos que podemos obtener de los datos preprocesados de cada una de las sesiones (seis modelos, puesto que se utilizan las tres primeras sesiones para generar los modelos y la cuarta sesión como experimento para obtener conclusiones de los pensamientos del usuario comparando sus mediciones con los modelos anteriores), obtener una predicción final a partir de la predicción más votada por cada uno de los seis modelos. Los modelos son obtenidos a partir de la combinación de dos sesiones, una sesión como aprendizaje y otro como ajuste de parámetros.

Estos scripts están diseñados para captar y procesar señales de encefalografía con el objetivo de predecir un total de tres estados: Reposo, movimiento de brazo derecho y movimiento de brazo izquierdo. Por tanto, se hace necesario realizar diversas modificaciones a dichas líneas de código con el fin de adaptar esta interfaz a nuestro fin.

3.1. Primera adaptación del algoritmo de adquisición de datos

Comenzaremos editando los scripts que realizan la adquisición de datos. Estos funcionan a partir de la visualización de un archivo ‘EEGRecordingPanel1.fig’ que hará de interfaz, por ello, se ha realizado una pequeña modificación de dicha figura, eliminando un panel correspondiente a la petición de mover el brazo derecho, y editar los textos de la figura, traduciéndolos al idioma español y adaptándolos a nuestras necesidades (Pensamiento de palabra aleatoria e intención de caminar), lo que incluye un renombramiento en las etiquetas de

algunos de los elementos presentes. En la Ilustración 12 se muestra la figura original y la figura ya modificada:

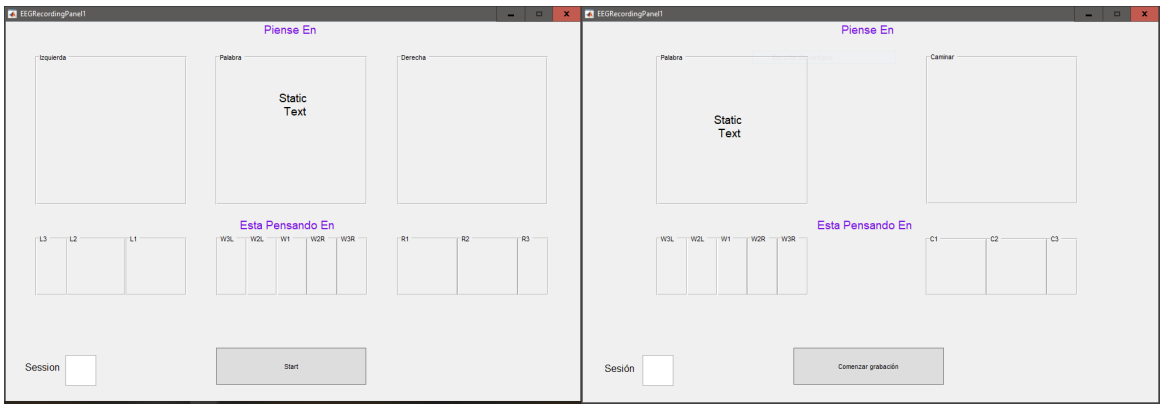

Ilustración 12. Interfaces de usuario. A la izquierda, la interfaz original; A la derecha, la interfaz ya editada para nuestro propósito

Por otro lado, deberemos editar el algoritmo 'EEGRecordingPanel1.m', el cual modifica la interfaz de usuario, y los correspondientes algoritmos invocados por esta función, que se encargan de adquirir y almacenar los estados y señales del kit EEG. Para ello, eliminaremos los 'set' del color de 'Right' y 'Left', e introduciremos un nuevo 'set' para 'Caminar', etiqueta que hemos colocado al recuadro de la nueva figura.

El algoritmo otorga a la variable 'status' tres puntuaciones: '3' para la intención de movimiento del brazo izquierdo, '5' para la intención de movimiento del brazo derecho, y '7' para el pensamiento de una palabra. En nuestro caso, al tener sólo dos estados, otorgaremos una puntuación de '1' para el pensamiento de una palabra aleatoria, y '2' para la intención de caminar. Para ello, realizaremos las siguientes modificaciones:

- La matriz 'status_set_matrix', inicializada en la línea 114 del algoritmo 'eeglogger_statusesBCIforms.m', contiene todos los estados que reproducirá la interfaz en cada sesión, con valores de 3, 5 y 7 ubicados de forma arbitraria. Hemos modificado esta matriz, modificando estos valores por unos y doses. Se ha contemplado la opción de utilizar algoritmos de generación aleatoria de números, pero se ha descartado tras valorar la posibilidad de respetar el formato de programación inicial.
- Dentro del algoritmo 'PresentStatus.m', hemos adaptado el código a los nuevos estados (1 y 2) y sus funcionalidades (cambio en el nombre de las etiquetas dentro de la figura 'EEGRecordingPanel1.fig').

- Dentro del algoritmo 'PresentEstimation.m', hemos adaptado el código de manera análoga al punto anterior, debido a los cambios sufridos en los estados y las etiquetas de la figura 'EEGRecordingPanel1.fig'.

El headset Emotiv EPOC incluye un giroscopio en la zona central de la diadema y permite la opción de leer los datos que obtiene el mismo de una manera similar a la lectura de datos de los sensores salinos. Hemos decidido añadir una funcionalidad nueva como apoyo al proceso de decisión de la intención de movimiento del usuario a través del giroscopio que integra el Headset. Para realizar lecturas del giroscopio que incorpora el Headset Emotiv Epoc 1.0, hemos realizado una serie de modificaciones en el algoritmo original desarrollado por el Dr. Juan Antonio Martínez León en su Tesis Doctoral [30]. Para ello, se ha optado por añadir dos columnas más a la matriz de datos obtenidos del Headset, llamados 'gyroX' y 'gyroY'. Además, sabemos que en la matriz 'output_matrix', donde se almacenan todos los datos recibidos por el Headset, los datos correspondientes al giroscopio se localizan en las columnas 18 y 19.

De esta manera, hemos decidido almacenar un nuevo dato de tipo .csv que almacene únicamente los estados, y los valores obtenidos en las variables 'gyroX' y 'gyroY'. Así mismo, el algoritmo queda de la siguiente manera:

```
fname_gyr = strcat(strcat('Exp',num2str(experiment_number)),'-
gyroscope.csv');
DataChannelsNamesExported = {'gyroX','gyroY'};
matrix_export = output_matrix(1:end_time,[26 18 19]);
cell2csv(fname_gyr,DataChannelsNamesExported,',');
dlmwrite(fname_gyr, matrix_export,'-append');
```

Una vez implementadas estas líneas, cada vez que se ejecute un ensayo para la grabación de datos, además del 'ExpX.csv' que contiene los datos de los sensores salinos, se generará otro .csv con los datos del giroscopio, llamado 'ExpX-gyroscope.csv', como se puede ver en la Ilustración 13 de los archivos generados. Más adelante se considerará el uso que se le puede dar a estos datos, a priori puede servir como refuerzo para definir una parada de emergencia que el sistema no esté detectando.

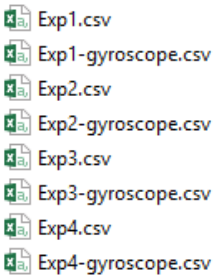

Ilustración 13. Ejemplo de archivos 'raw' generados por el algoritmo

3.2. Adquisición de datos y conclusiones con la primera versión del algoritmo

Realizamos varias rondas de ensayos con el equipo. Para ello, ejecutamos el algoritmo 'EEGRecordingPanel1.m' y, en la ventana que aparece, introducimos el número de sesión a grabar. Realizaremos, por cada ensayo, cuatro sesiones que generan cuatro archivos 'ExpN.csv' como podemos ver en la Ilustración 13.

A continuación, usamos un algoritmo de procesado de datos que genera, a partir de los tres primeros ensayos, modelos de predicción los cuales serán testados a partir del cuarto ensayo, intentando predecir el estado 'marcha' o 'paro' que desea el usuario. De esta manera podemos obtener tres porcentajes de acierto y una tabla resumen del número de aciertos y fallos a la hora de inferir 'marcha' o 'paro'. Esta etapa será desarrollada en profundidad en el Apartado 4.

Los primeros experimentos se realizan con una duración de 150 segundos y 15 segundos entre estados, creando así 10 estados diferentes y un total, entre las cuatro sesiones, de 10 minutos de experimento. Como podemos observar en la Ilustración 14, los resultados no fueron los deseados:

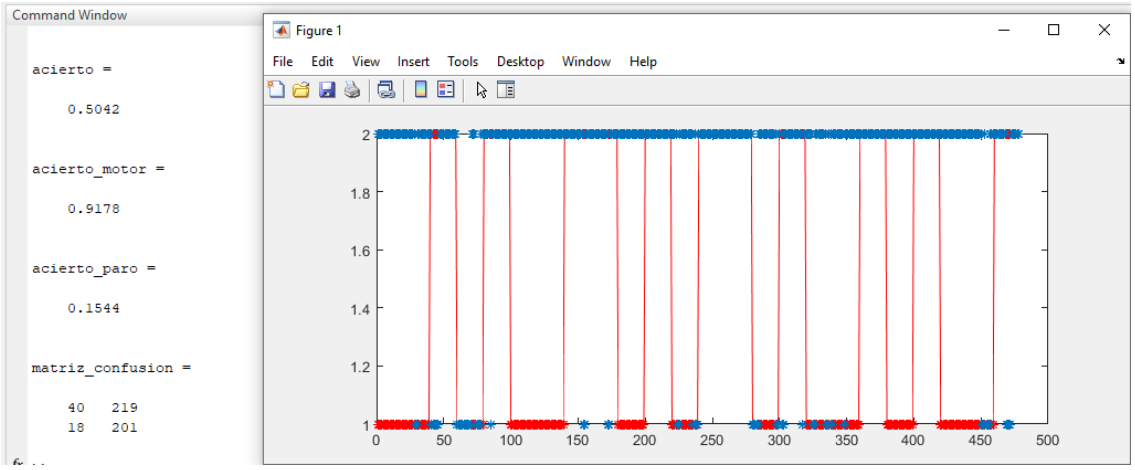

Ilustración 14. Resultado de un experimento dentro de la primera tanda de experimentos

El experimento anterior es uno de los varios que se realizaron para descartar errores como pueden ser faltas de concentración, pérdidas de conexión entre el equipo y el software, errores de lectura del equipo, etc. Todos los ensayos reflejaban, aproximadamente, una tasa de acierto en torno al 50%. Cabe destacar que, en términos probabilísticos, el experimento que estamos abordando sería de tipo dicotómico equilibrado, es decir, que la probabilidad de definir

aleatoriamente un estado y acertar es del 50%. Por tanto, nos estamos aproximados a resultados aleatorios.

Esto es debido, como podemos ver en la gráfica, a que el software de procesado considera que el usuario desea en prácticamente todo momento comenzar a caminar, cuando eso realmente no es así. Es, por eso, que el porcentaje de acierto de marcha 'acierto_motor' es tan alto y el de paro 'acierto_paro' tan bajo. Además, podemos observar en 'matriz_confusion' que sólo acertó 40 veces el paro, fallando 219 veces y, por el contrario, dio 201 órdenes de marcha, fallando 18 veces. Se hace necesario, por tanto, adoptar medidas de ajuste en los experimentos.

3.3. Modificación de las características del experimento

Dado que los usuarios testados recalcaron llegar mentalmente cansados a la última sesión del experimento y al final de todas las sesiones, se consideró la opción de acortar los tiempos del experimento. Para ello, se redujo la ventana entre estado y estado de 15 a 5 segundos y se mantuvo el número de estados por sesión en 10, reduciendo así en total las sesiones a 50 segundos y el experimento completo a 3 minutos 20 segundos. Tras realizar distintos experimentos, los resultados obtenidos en todos los experimentos se asemejan a los obtenidos concretamente en la Ilustración 15:

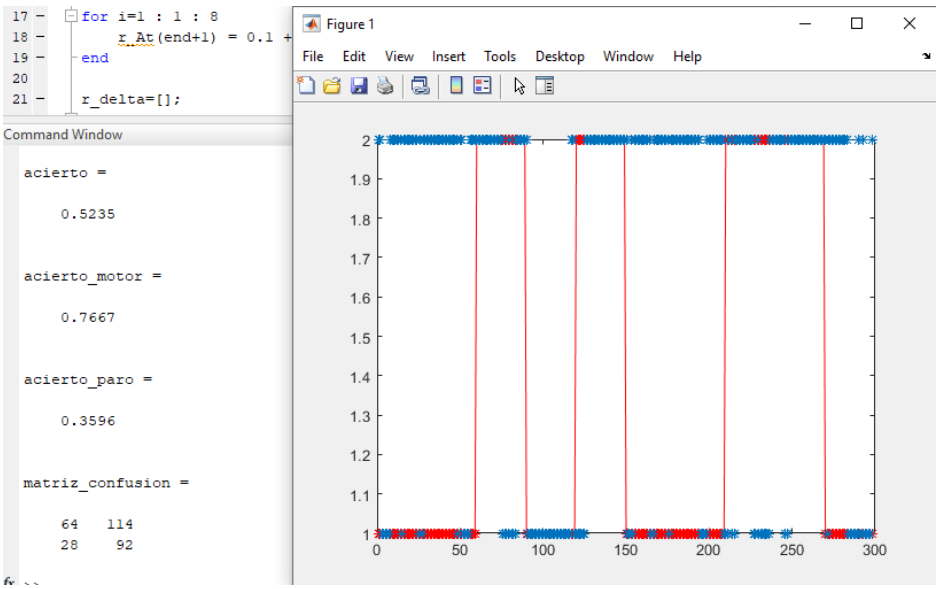

Ilustración 15. Resultado de un experimento dentro de los experimentos realizados bajo las nuevas características

Como podemos observar, el porcentaje de acierto sigue siendo sensiblemente bajo, en torno al 52%, aunque hemos logrado reducir el efecto de monopolio del estado 'marcha' respecto al 'paro' que veíamos en la primera tanda de experimentos.

Se hace necesario un estudio exhaustivo del causante de tanto error en la elección del estado y, observando los datos 'raw' almacenados en los 'ExpN.csv', podemos ver claramente cómo la columna de tiempos de adquisición contiene valores extraños. Como es conocido, la frecuencia de adquisición del equipo es de $f = 128\,Hz$, por lo que deberíamos obtener datos cada $T = \frac{1}{128} = 0.0078\,s$. Sin embargo, podemos observar en los datos 'raw' lo siguiente: (Véase la Ilustración 16)

L	M	N	O	P
ED_FC6	ED_F4	ED_F8	ED_AF4	ED_ES_TIM...
Number	Number	Number	Number	Number
ED_FC6	ED_F4	ED_F8	ED_AF4	ED_ES_TIM...
669.23	2714.4	4046.7	4185.6	0.039096
668.21	2716.4	4041	4182.1	0.039096
667.18	2718.5	4044.6	4185.6	0.039096
668.72	2715.9	4054.9	4192.3	0.039096
670.26	2714.4	4048.7	4192.8	0.039096
670.26	2715.4	4052.8	4190.3	0.039096
669.74	2716.9	4050.3	4187.2	0.039096
669.74	2717.4	4051.3	4187.7	0.039096
668.72	2716.9	4060	4190.3	0.039096
669.74	2716.9	4048.2	4188.2	0.039096
671.28	2716.4	4051.3	4183.6	0.039096
668.72	2716.4	4055.4	4184.6	0.039096
668.21	2717.4	4051.3	4187.7	0.039096
669.74	2716.4	4054.9	4185.6	0.039096
670.26	2716.4	4049.7	4185.1	0.039096
670.26	2718.5	4058.5	4186.7	0.039096
668.72	2718.5	4065.1	4186.7	0.039096
669.23	2716.9	4059	4188.2	0.039096
671.28	2716.9	4065.1	4187.2	0.039096
670.77	2719.5	4056.9	4185.6	0.039096
670.26	2717.4	4057.4	4186.7	

Ilustración 16. Columna de tiempos que genera el equipo EEG Emotiv Epoc

Si graficamos esa columna en el tiempo, obtenemos lo siguiente: (Veéase la Ilustración 17)

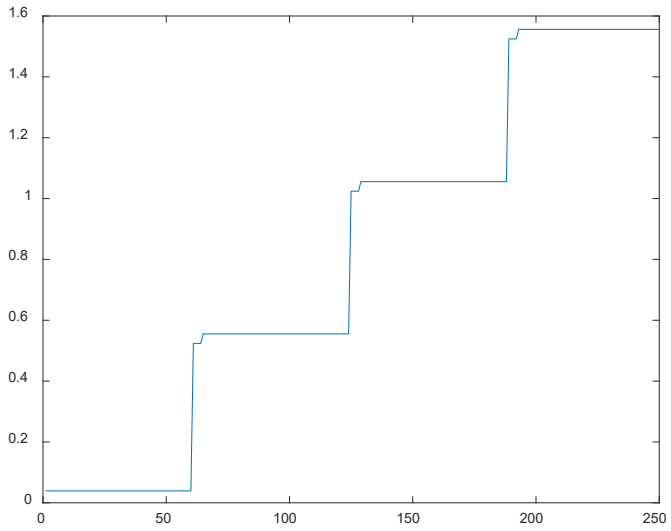

Ilustración 17. Gráfica temporal de los primeros 250 datos del experimento anterior. En el eje de abscisas se representa el número de muestra correspondiente. En el eje de ordenadas, el valor del tiempo de adquisición que le corresponde.

Se puede observar, dada la representación de la Ilustración 17, que las muestras adoptan durante un período el mismo tiempo de adquisición, mientras que el resultado esperado sería que dicho tiempo de adquisición creciera de manera lineal a razón de $\frac{1}{128}$ segundos por muestra. Dado que el preprocesado de los datos antes de la generación del modelo de predicción incluye un tratado mediante Densidad Espectral de Potencia (PSD), el cual es una función matemática basada en la frecuencia de muestreo anteriormente citada, el hecho de obtener patrones inusuales en estos tiempos de adquisición hace que obtengamos malos resultados en este punto del procesado de datos.

3.4. Corrección de tiempos de adquisición

Hemos visto anteriormente que el tiempo de adquisición que retornaba el equipo no cumplía con las condiciones necesarias para hacer la PSD. Por ello, hemos buscado el lugar donde se genera ese error. El algoritmo de adquisición originalmente desarrollado por [30] contiene una sección donde guarda el volcado de datos que devuelve el equipo cuando se le solicitan los datos a través de la librería de EMOTIV:

```
if (nSamplesTaken ~= 0)
   data = libpointer('doublePtr',zeros(1,nSamplesTaken));
   for i= 1:length(fieldnames(enuminfo.EE_DataChannels_enum))
     calllib('edk','EE_DataGet',hData,
DataChannels.([DataChannelsNames{i}]), data, uint32(nSamplesTaken));
     data_value = get(data,'value');
     output_matrix(cnt+1:cnt+length(data_value),i) = data_value;
   end
```

```
    %(JML) - New column on output matrix for status
    output_matrix(cnt+1:cnt+length(data_value),i+1) = status;
    cnt = cnt + length(data_value);
end
```

Se ha comprobado de manera experimental que este volcado no tiene por qué ser de fila en fila, sino que puede tener varias filas a la vez, todas con el mismo tiempo de adquisición retornado en 'ED_ES_TIMESTAMP'. Para solucionar esto, se han hecho uso de las funciones de MATLAB 'tic' y 'toc' que devuelven la Δt desde que se ejecuta el 'tic' hasta que se ejecuta el 'toc' y se han utilizado como tiempos de adquisición, en sustitución de 'ED_ES_TIMESTAMP'. Posteriormente, en la parte del código donde se organizan los datos 'raw' antes de almacenarlos en los ficheros '.csv', hemos introducido las siguientes líneas de código:

```
%%%%Añadido para tratar tiempos
    tiempo=toc;
…
…
…
    if cnt==0
        escalon=tiempo/length(data_value);
        for j=1:length(data_value)
            output_matrix(cnt+j,i+2)=(j-1)*escalon;
        end
    else
        tiempo_anterior=output_matrix(cnt,i+2);
        escalon=(tiempo-tiempo_anterior)/length(data_value);
        for j=1:length(data_value)
            output_matrix(cnt+j,i+2)=tiempo_anterior+j*escalon;
        end
    end
```

Dado que el volcado sigue sin ser de fila en fila, el uso de 'tic' y 'toc' no evita la aparición de varias filas con el mismo tiempo de adquisición, por lo que se ha optado por dividir la delta de tiempo entre dos volcados distintos entre el número de filas que tiene el volcado analizado, para así obtener el incremento de tiempo que debería haber entre fila y fila, consiguiendo así ese $T = 0.0078\ s$ calculado anteriormente.

Por último, resta dejar de almacenar los valores obtenidos del casco y sustituirlos por los valores obtenidos mediante el uso de 'tic' y 'toc':

```
matrix_export = output_matrix(1:end_time,[26 4:17 27]);
```

De esta manera, hemos solventado estos errores que observamos en el Apartado 3.3. Modificación de las características del experimento, como podemos observar en la Ilustración

18, donde se destaca la linealidad en las etiquetas temporales frente a la Ilustración 17, donde iba experimentando ciertos escalones de tiempo.

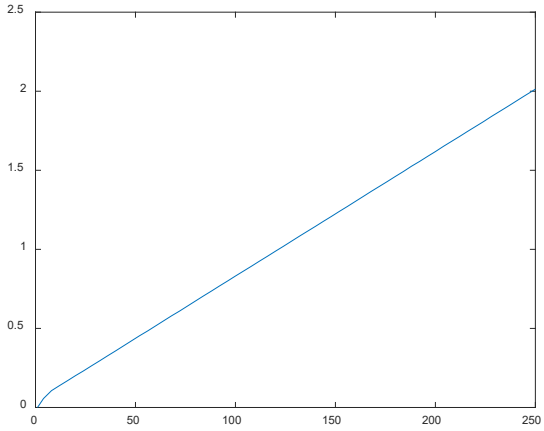

Ilustración 18. Gráfica de las 250 primeras mediciones de tiempo en un experimento. En el eje de abscisas se representa el número de muestra, mientras que en el eje de ordenadas se representa el tiempo de adquisición correspondiente.

Nuevamente se realizan otras rondas de ensayos, ejecutando posteriormente los algoritmos de procesado de datos para obtención de conclusiones. Se ha mantenido el formato adoptado en el Apartado 3.3., consistente en 4 experimentos, cada uno con 15 estados distintos de 5 segundos cada uno. El resultado se muestra en la Ilustración 19:

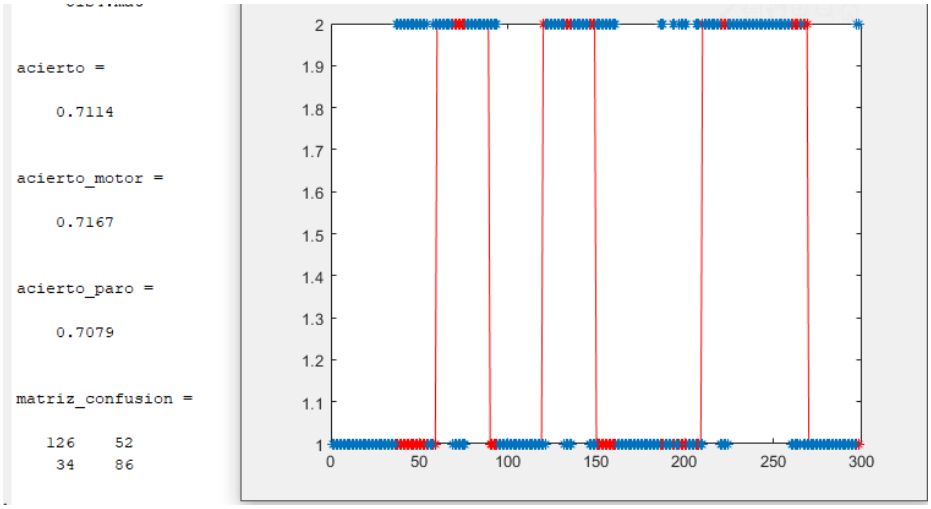

Ilustración 19. Resultados para el experimento realizado con todas las modificaciones en los algoritmos

Podemos observar que el porcentaje de acierto global ha subido considerablemente frente a otros experimentos como el mostrado en la Ilustración 15. Además, el porcentaje de acierto tanto de motor como de paro es muy similar al porcentaje de acierto global, solventando así el problema de que el control *Supervised dFasArt* infiera constantemente el mismo estado (como se ha podido ver en otros experimentos, solía inferir de manera constante el estado de marcha). Podemos concluir también que, pese a que un 70% de acierto global no sea un resultado muy

elevado, vemos en la gráfica de la Ilustración 15 que estos errores se asocian a situaciones muy concretas:

- Cuando el estado se hace más largo de lo habitual, en los instantes finales suelen ocurrir más fallos de lo normal. Esto puede ser debido a errores de concentración del usuario, como venía ocurriendo en los experimentos iniciales con estados de 15 segundos, que son malinterpretados o interpretados de manera aleatoria por el procesado.
- En las transiciones marcha-paro o viceversa existe un Δt en los que se suele inferir el estado previo. Esto puede ser debido al tiempo que tarda el usuario en concentrarse en el nuevo estado una vez se muestra por pantalla este cambio de estado.

Si obviáramos estas secciones donde se acumulan la mayoría de fallos podemos concluir que tenemos un sistema con una robustez elevada, teniendo en cuenta los objetivos del Proyecto que implementa un hardware de bajo coste en comparación con otros *Headstes* disponibles en el mercado. El algoritmo de obtención de datos, con las modificaciones pertinentes, se puede ver en su totalidad en el Anexo I.

4. Generación de modelos de predicción y estimación de estados del pensamiento

Como hemos comentado en el Apartado 3, el proceso que comprende los pasos desde la grabación de los ensayos hasta la obtención de conclusiones, abarca los siguientes pasos:

- Filtro Laplaciano de los datos: Se emplea un filtro de tipo Laplaciano (Véase el Apartado 2.3.2. Tratamiento de señales EEG) para la eliminación de ruidos y señales no deseadas. Para ello, se hace uso de un archivo de posiciones en el plano cenital de los sensores según el esquema 10-20 para EEG, tal y como se muestra en la Ilustración 20:

```
x =
  Columns 1 through 14
   -0.1501         0    0.1501   -0.1785    0.1785   -0.5439   -0.4200   -0.2961   -0.1480         0    0.1480    0.2961    0.4200    0.5439
  Columns 15 through 28
   -0.7000   -0.5404   -0.3613   -0.1821         0    0.1821    0.3613    0.5404    0.7000   -0.7692   -0.5769   -0.3846   -0.1923         0
  Columns 29 through 42
    0.1923    0.3846    0.5769    0.7692   -0.8400   -0.7000   -0.5404   -0.3613   -0.1821         0    0.1821    0.3613    0.5404    0.7000
  Columns 43 through 56
    0.8400   -0.5439   -0.4200   -0.2961   -0.1480         0    0.1480    0.2961    0.4200    0.5439   -0.4187   -0.2961   -0.1785         0
  Columns 57 through 64
    0.1785    0.2961    0.4187   -0.1785   -0.1501         0    0.1501    0.1785

y =
  Columns 1 through 14
    0.7545    0.7545    0.7545    0.6094    0.6094    0.5439    0.4813    0.4187    0.4017    0.3846    0.4017    0.4187    0.4813    0.5439
  Columns 15 through 28
    0.2800    0.2423    0.2197    0.1971    0.1971    0.1971    0.2197    0.2423    0.2800         0         0         0         0         0
  Columns 29 through 42
         0         0         0         0   -0.3800   -0.2800   -0.2423   -0.2197   -0.1971   -0.1971   -0.1971   -0.2197   -0.2423   -0.2800
  Columns 43 through 56
   -0.3800   -0.5439   -0.4813   -0.4187   -0.4017   -0.3846   -0.4017   -0.4187   -0.4813   -0.5439   -0.6893   -0.6494   -0.6094   -0.6094
  Columns 57 through 64
   -0.6094   -0.6494   -0.6893   -0.9022   -0.7545   -0.7692   -0.7545   -0.9022
```

Ilustración 20. Valores almacenados en los arrays 'x' e 'y' para el archivo 'electrodes.mat'

Si representamos dichos datos en unos ejes coordenados, podemos ver que el resultado obtenido (véase la Ilustración 21) es el esquema 10-20 que abordamos en el Apartado 2.3.1:

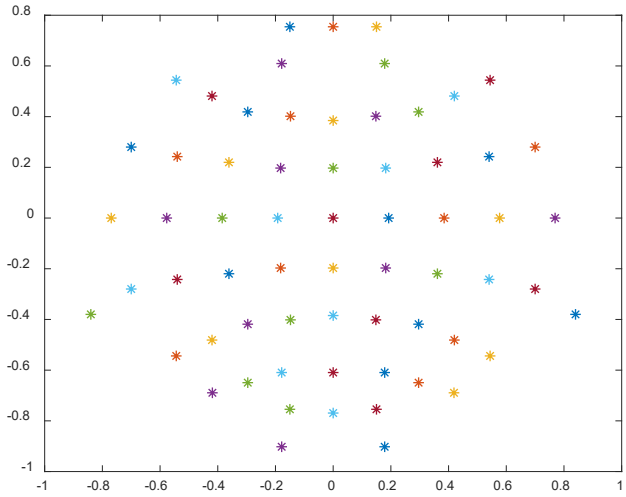

Ilustración 21. Representación gráfica de las coordenadas 'x' e 'y' que contiene el archivo 'electrodes.mat'

- Densidad espectral de potencia (PSD) de cada uno de los sensores, con una ventana de un segundo y para frecuencias comprendidas entre 8 y 30 Hz con saltos de 2 Hz, procesando así un total de 12 frecuencias (véase la Ilustración 22). La aplicación de PSD en EEG es una técnica computacional muy extendida, ya que permite obtener un modelo autorregresivo paramétrico, el cual nos podrá ofrecer información en función de la potencia de la señal para cada frecuencia. Dicho de otro modo, ciertos patrones de potencia de señales nos indicarán que un suceso está ocurriendo en la EEG.

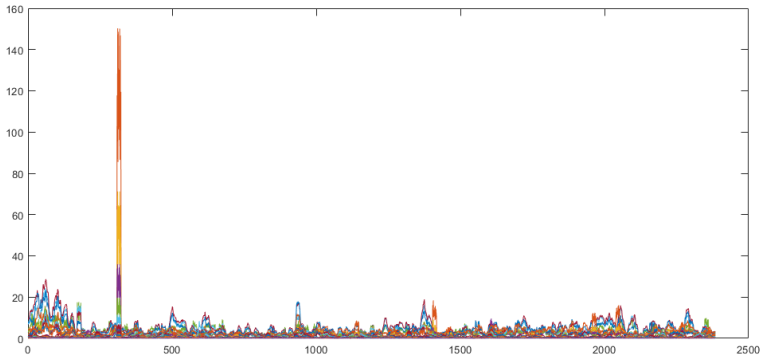

Ilustración 22. PSD de un ensayo realizado con el equipo Emotiv EPOC durante el desarrollo de este Proyecto

Este método numérico se puede hacer gracias al comando ‘pwelch’ de MATLAB, el cual implementa una PSD dados una serie de datos, una ventana, una serie de frecuencias y una frecuencia de muestreo (en nuestro caso 128 Hz, que es la frecuencia de muestreo del equipo).

- Generación de modelos de aprendizaje: Esta etapa comprende todas las labores matemáticas para generar los modelos de tipo *supervised dFasArt* y todos los parámetros que lo componen. Para ello, a partir de cada una de las tres sesiones previamentes cargadas, se genera el modelo correspondiente y se inicializan sus parámetros.
- Optimización de parámetros de los modelos de aprendizaje: A partir de la comparación entre dos modelos generados, se optimizan los parámetros de los mismos para que incrementen su tasa de acierto. Al compararse dos a dos, entre tres modelos, al final resultan un total de seis combinaciones: Modelos 1 y 2, modelos 1 y 3, modelos 2 y 1, modelos 2 y 3, modelos 3 y 1, modelos 3 y 2.
- Generación de estados: A partir de la sesión 4 y usando todos los modelos generados anteriormente, se infieren los estados que el EEG ha leído. Para ello, se leen los datos de la sesión 4, se hacen pasar por los seis modelos generados y se obtienen seis conclusiones. Finalmente, el estado modal entre los seis resultados será el definitivo.

Todos estos pasos están integrados en un algoritmo llamado 'batch.m' el cual ha sido facilitado por los directores del Proyecto y que requiere únicamente de incluir en la carpeta contenedora los cuatro ensayos en bruto obtenidos en los algoritmos de adquisición de datos (véase la Ilustración 23). Además, este algoritmo facilita una serie de estadísticas finales para poder ponderar los resultados obtenidos.

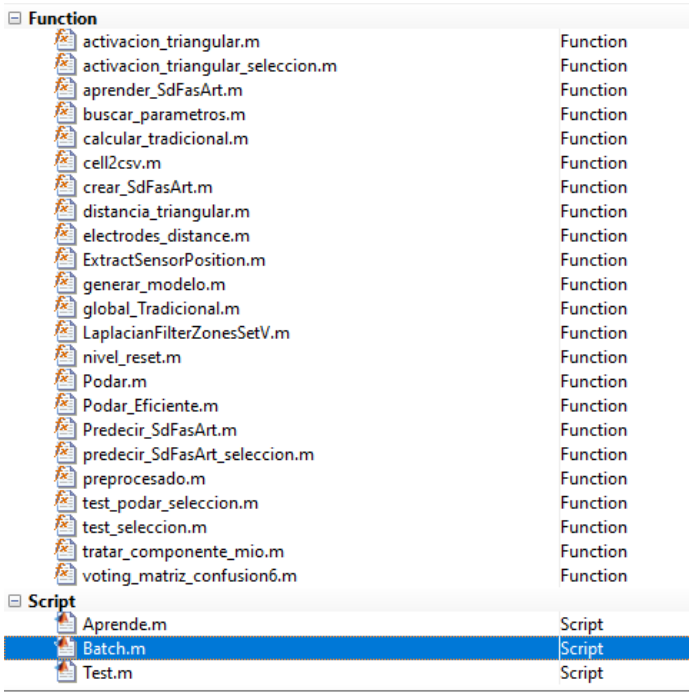

Ilustración 23. Script 'batch.m' y funciones que acompañan al mismo

4.1. Experimentos realizados

Para lograr el sistema BCI más robusto posible, se han realizado diferentes experimentos con el fin de encontrar la combinación de acciones que mayor tasa de acierto ofrezca. Los tipos de experimentos realizados son los siguientes:

1. **Pensamiento en paro**: Palabra aleatoria dada una inicial
   Pensamiento en marcha: Imaginación de proceso de caminar
2. **Pensamiento en paro**: Palabra aleatoria dada una inicial
   Pensamiento en marcha: Imaginación de proceso de caminar acompañado de pasos reales
3. **Pensamiento en paro**: Palabra aleatoria dada una inicial
   Pensamiento en marcha: Imaginación del proceso de caminar y leve movimiento oscilante de las piernas (sentado)
4. **Pensamiento en paro**: Mente en blanco
   Pensamiento en marcha: Imaginación del proceso de caminar y leve movimiento oscilante de las piernas (sentado)
5. **Pensamiento en paro**: Tablas de multiplicar
   Pensamiento en marcha: Imaginación del proceso de caminar y leve movimiento oscilante de las piernas (sentado)

Como hemos comentado, el algoritmo 'batch.m' implementa un conteo de aciertos y un porcentaje de aciertos, además de una matriz de confusión que representa los aciertos y fallos para cada uno de los estados de marcha y paro. A continuación, se desarrolla una tabla resumen con los distintos experimentos realizados y sus porcentajes de acierto medios (Tabla 1)

Tabla 1. Resultados de experimentos según el tipo de marcha y paro

Tipo de experimento	Número de ensayos	Acierto global (%)	Aciertos marcha	Fallos marcha	Aciertos paro	Fallos paro
1	7	53.65%	201.29	17.71	39.85	219.15
2	2	51.46%	195.5	23.5	35.5	224.5
3	4	62.62%	108.5	11.5	62	116
4	2	59.56%	103.5	16.5	58.5	119.5
5	5	72.13%	90.4	29.6	122.7	55.3

Podemos observar que los ensayos 1, 2 y 3 se caracterizan, además de por tener un porcentaje de acierto global relativamente bajo, por predecir en exceso un estado de marcha (véase la Ilustración 24). Esto lo podemos ver en sus cantidades de aciertos en marcha respecto a paro (rondando un porcentaje de acierto en torno al 90%) y un porcentaje de acierto de paro excesivamente bajo (En torno al 15%).

Ilustración 24. Resultados para un ensayo de tipo 1, donde se puede ver, con asteriscos azules, que el sistema siempre está estimando 'marcha'

No es hasta el experimento 4 en menor medida, y el 5 en mayor medida, cuando logramos observar cambios respecto a este comportamiento. En un ensayo del experimento 54 podemos comprobar que el sistema ya no infiere constantemente estados de marcha, sino que alterna con mayor frecuencia. En el ensayo 5, además, podemos observar que lo realiza con bastante fidelidad respecto al estado real que se solicitó al usuario del ensayo (véase la Ilustración 25).

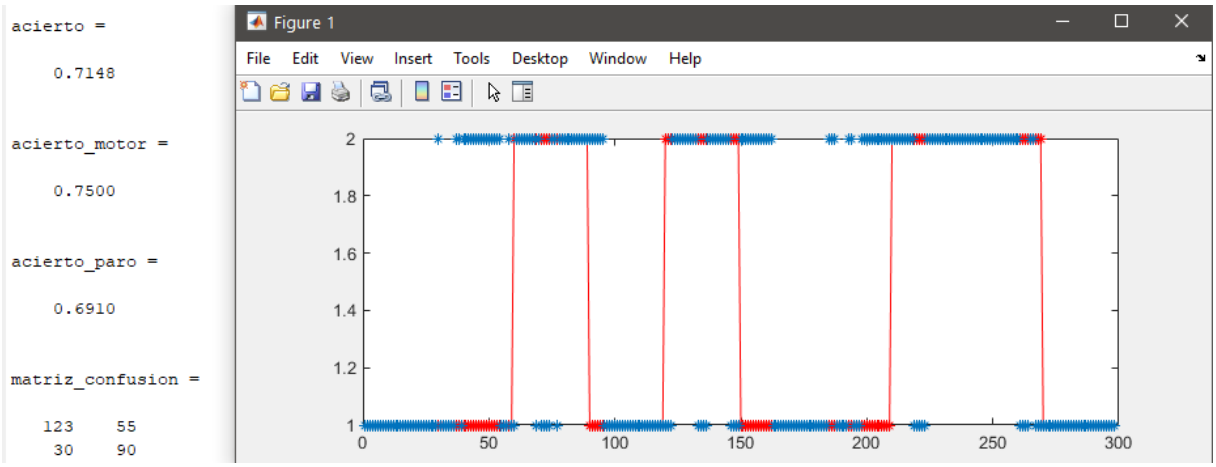

Ilustración 25. Resultados para un ensayo de tipo 5

Un ejemplo de ensayo de tipo 5 lo vemos en la Ilustración 25. En ella se pueden ver fenómenos similares a los ya comentados al final del Apartado 3.4. Corrección de tiempos de adquisición, donde se empleó un experimento de este tipo para corregir el problema de los escalones temporales que devolvía el equipo EEG:

- Cuando el estado se hace más largo de lo habitual, en los instantes finales suelen ocurrir más fallos de lo normal. Esto puede ser debido a errores de concentración del usuario, como venía ocurriendo en los experimentos iniciales con estados de 15 segundos, que son malinterpretados o interpretados de manera aleatoria por el procesado.
- En las transiciones marcha-paro o viceversa existe un Δt en los que se suele inferir el estado previo. Esto puede ser debido al tiempo que tarda el usuario en concentrarse en el nuevo estado una vez se muestra por pantalla este cambio de estado.

4.2. Peso de cada sensor en la toma de decisiones

Es interesante realizar un estudio acerca de qué sensores aportan más y aportan menos en la toma de decisiones para los modelos. De esta manera, podemos reducir el coste computacional a la hora de realizar predicciones en tiempo real, puesto que será la etapa más exigente a nivel de potencia necesaria. Para ello, una manera de obtener conclusiones por sensor sería ajustar el parámetro $A_R = 0$. De esta manera, se van a generar únicamente dos reglas por modelo. Los pasos a hacer en el algoritmo 'batch.m' y sus funciones invocadas son las que se pueden ver en las siguientes Ilustración 26, Ilustración 27 e Ilustración 28, a partir de las posiciones de los sensores mostradas en la Tabla 2:

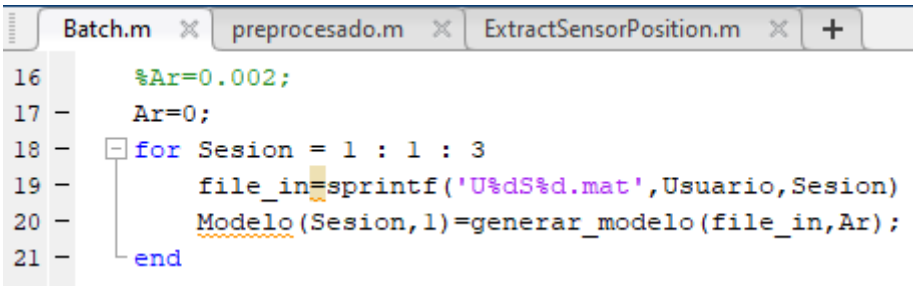

Ilustración 26. Ajuste de Ar en 'batch.m'. Por defecto el parámetro viene fijado en 0.002. De esta manera, lograremos que se generen dos únicas reglas por modelo.

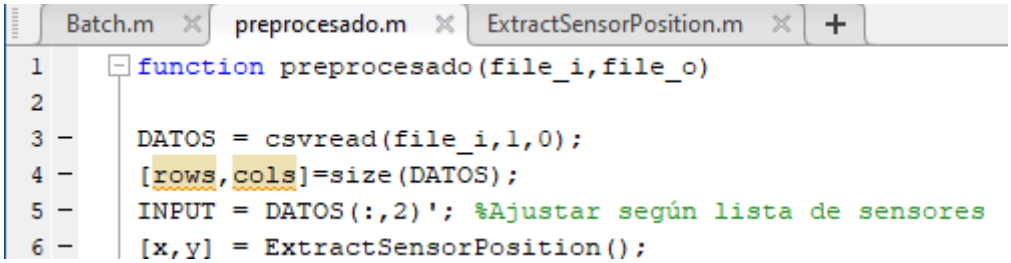

Ilustración 27. Selección del sensor a estudiar según la lista de sensores (Véase Tabla 2) en la función 'preprocesado.m'

Tabla 2. Posición de los sensores en la columna de datos en bruto

Nombre del sensor	Columna correspondiente	Nombre del sensor	Columna correspondiente
AF3	2	F7	3
F3	4	FC5	5
T7	6	P7	7
O1	8	O2	9
P8	10	T8	11
FC6	12	F4	13
F8	14	AF4	15

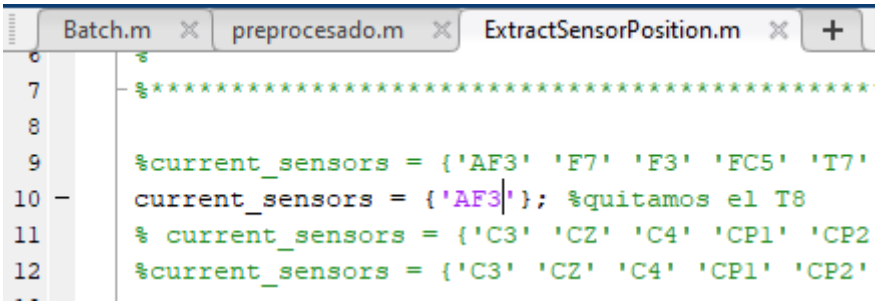

Ilustración 28. Selección del sensor a estudiar en la función 'ExtractSensorPosition.m'

Realizando estos ajustes podremos obtener conclusiones acerca de lo que cada sensor categoriza. Por ejemplo, tras el ajuste visto en la Ilustración 26, Ilustración 27 e Ilustración 28, donde se ha ajustado el algoritmo para obtener conclusiones del sensor 'AF3', una ejecución del algoritmo 'batch.m' nos permitirá concluir que dicho sensor, en general, sólo categoriza estados de 'marcha' independientemente del estado real en el que se encuentre (véase Ilustración 29).

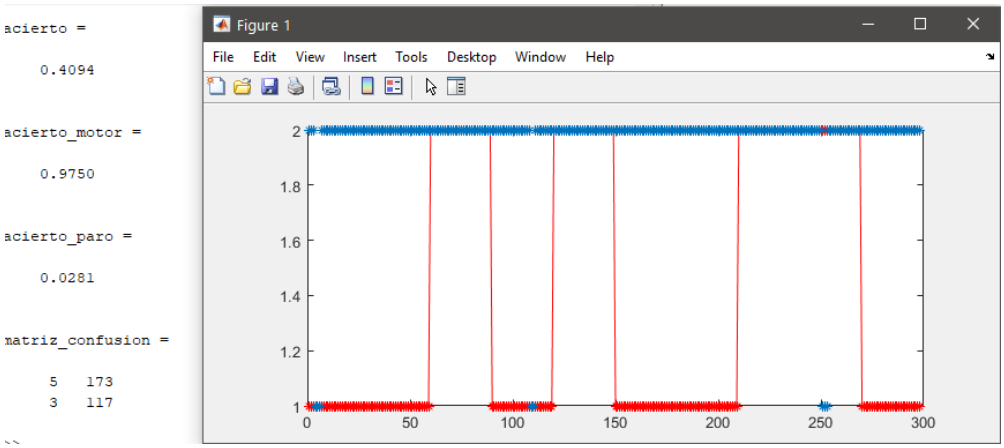

Ilustración 29. Reslutados para el sensor 'AF3'

Realizando la misma labor para todos los sensores vistos en la Tabla 2, obtenemos las siguientes conclusiones (véase Tabla 3)

Tabla 3. Resultados para cada sensor utilizando sólo dos etiquetas (Ar=0)

Sensor	Características	Acierto (%)	Aciertos paro	Fallos paro	Aciertos marcha	Paros marcha
AF3	1-12	40,94%	5	173	117	3
F7	13-24	40,60%	1	177	120	0
F3	25-36	46,98%	48	130	92	28
FC5	37-48	38,26%	20	158	94	26
T7	49-60	40,94%	2	176	120	0
P7	61-72	45,30%	20	158	115	5
O1	73-84	62,42%	78	100	108	12
O2	85-96	67,45%	152	26	49	71
P8	97-108	50,67%	91	87	60	60
T8	109-120	56,38%	114	64	54	66
FC6	121-132	61,74%	121	57	63	57
F4	133-144	60,40%	91	87	89	31
F8	145-156	52,01%	111	67	44	76
AF4	157-168	41,61%	7	171	117	3

De la tabla anterior podemos dividir lrEos sensores en

- Sensores que categorizan sólo marcha: AF3, F7, T7, P7, AF4, FC5
- Sensores que categorizan ambos estados, pero con bajo porcentaje de acierto: P8, F8, F3, T8
- Sensores que tienen buenos porcentajes globales de acierto: O2, O1, FC6, F4

Tras estas conclusiones, podemos comparar el porcentaje de acierto, una vez el parámetro Ar haya sido reajustado a 0.002 como lo estaba inicialmente, usando grupos concretos de

sensores. Así, los resultados con distintas combinaciones de sensores se muestran en la Ilustración 30, Ilustración 31, Ilustración 32, Ilustración 33 e Ilustración 34:

```
acierto =

    0.7114


acierto_motor =

    0.7167


acierto_paro =

    0.7079


matriz_confusion =

   126    52
    34    86
```

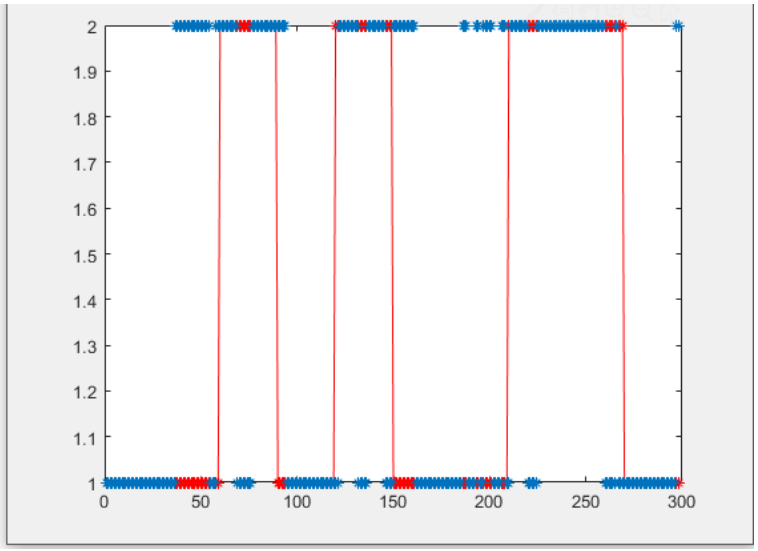

Ilustración 30. Resultados con todos los sensores. Ar=0.002

```
acierto =

    0.6913


acierto_motor =

    0.7500


acierto_paro =

    0.6517


matriz_confusion =

   116    62
    30    90
```

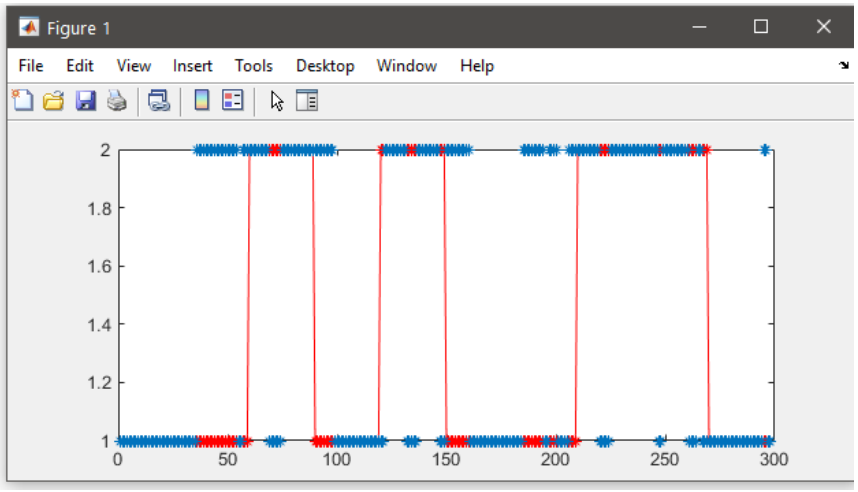

Ilustración 31. Resultados con los sensores O1, O2, FC6 y F4

```
acierto =

    0.7114


acierto_motor =

    0.8083


acierto_paro =

    0.6461


matriz_confusion =

   115    63
    23    97
```

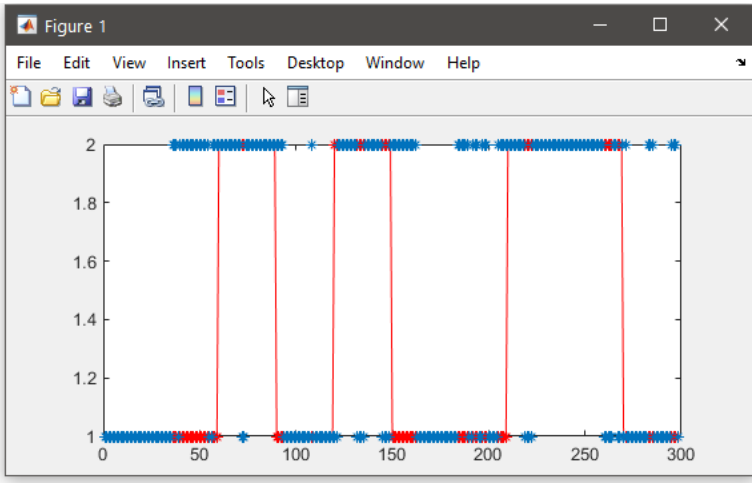

Ilustración 32. Resultados con sensores O1, O2 y F4. Ar=0.002

```
acierto =

    0.7081


acierto_motor =

    0.7583


acierto_paro =

    0.6742


matriz_confusion =

   120    58
    29    91
```

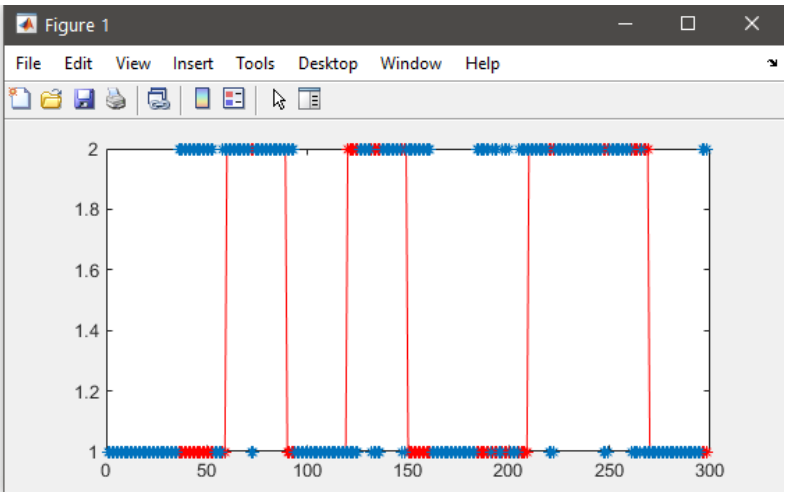

Ilustración 33. Resultados con sensores O1, O2 y FC6. Ar=0.002

```
acierto =

    0.7148


acierto_motor =

    0.7500


acierto_paro =

    0.6910


matriz_confusion =

   123    55
    30    90
```

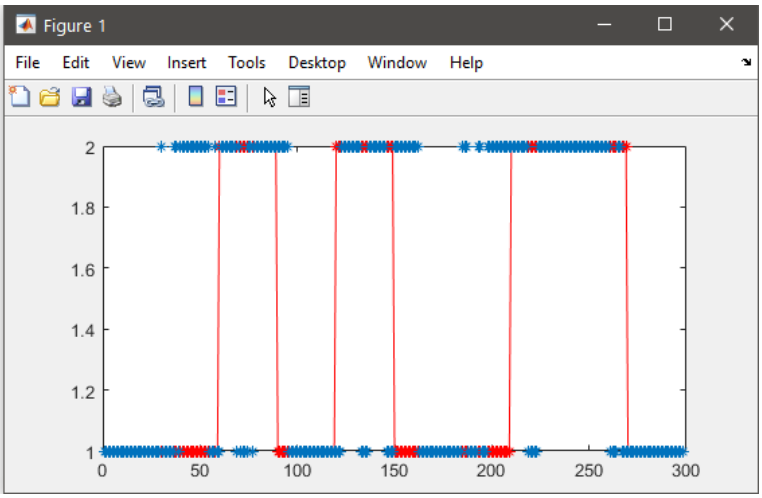

Ilustración 34. Resultados con O1 y O2. Ar=0.002

Podemos observar que usar los sensores O1 y O2 (véase la Ilustración 34) genera un mejor acierto (71.48% de acierto frente a 71.14% con todos los sensores), sin embargo, también categoriza más el estado ‘marcha’ respecto a los resultados originales, cosa que se ve reflejada

en la matriz de confusión (90 estados ‘marcha’ con los dos sensores contra 86 estados ‘marcha’ con todos los sensores). En conclusión, tenemos la posibilidad de reducir el número de sensores a dos si fuera necesario reducir la potencia necesaria para ejecutar el sistema, pero de cara a la viabilidad del Proyecto interesa más tener un porcentaje de acierto de ‘marcha’ y ‘paro’ cercano y similar al acierto global.

4.3. Conclusiones de los experimentos

Es muy complejo elaborar conclusiones a partir de datos EEG puesto que predecir a partir de una señal EEG es un proceso de comparación frente a otras señales guardadas anteriormente. No es posible estimar por qué unos tipos de ensayos generan más tasa de acierto que otros, pero hay ciertas hipótesis que podemos plantear:

- El hecho de que los ensayos tipo 4 y 5 son ensayos realizados en la etapa final del Proyecto, una vez que el usuario ha adquirido más experiencia con los ensayos EEG. Existen otros Proyectos que implementan EEG y que han experimentado elevadas diferencias en las tasas de acierto comparando entre usuarios experimentados y neófitos.
- Las señales EEG pueden presentar menores diferencias características en los ensayos tipo 1, 2 y 3 entre los estados marcha y paro, que en los tipos 4 y 5. De esta manera al sistema le resulta más sencillo clasificar las señales en ‘marcha’ o ‘paro’ teniendo unas señales EEG más ‘esclarecedoras’.
- Los ensayos de tipo 4 y 5 tienen correcciones de software implementadas según se avanzaba a lo largo del Apartado 3. Es, por ello, que esas mejoras en el software pueden generar pequeños cambios en los porcentajes de acierto.

5. Simulador 3D de exoesqueleto

El objetivo principal del Proyecto era la utilización de un equipo EEG como el Emotiv EPOC para el control mediante BCI de un exoesqueleto para el tren inferior. Sin embargo, dado que el trabajo fue desarrollado en los años 2020 y 2021, entre las dificultades que introdujo la pandemia por SARS-CoV-2 estuvo la imposibilidad de acceso a los Laboratorios de Investigación, ubicados en el Dpto. de Automática, Ingeniería Eléctrica y Tecnología Electrónica de la ETSII UPCT, lugar donde se encontraba el exoesqueleto EXO-LEGS.

Es, por ello, por lo que se ha optado por la opción de sustituir el exoesqueleto real por un simulador básico en 3D que represente lo que se quería implementar en el laboratorio. El simulador trabajará de la misma manera que lo haría el exoesqueleto original: A partir de un *array* de ángulos para las seis articulaciones del exoesqueleto (dos caderas, dos rodillas y dos tobillos) que se reproducirá en el tiempo, generando así un paso. En la Ilustración 35 se muestra una imagen del exoesqueleto en cuestión, con los ángulos de la pierna izquierda marcados en color verde:

Ilustración 35. Exoesqueleto disponible en el Dpto. Automática, Ingeniería Eléctrica y Tecnología Electrónica (UPCT). Sobre el mismo, dibujadas las barras (en rojo) y marcadas los ángulos de cada articulación (en verde)

Dichos ángulos se han obtenido de un Proyecto acerca del estudio de un paso estándar donde, a partir de un equipo conectado a las dos piernas y apoyados en una serie de potenciómetros y una tarjeta de adquisición de datos, se registraron y normalizaron todos los

ángulos que adquieren las articulaciones de la cadera, rodilla y tobillo en las dos piernas de varias personas, realizado por el alumno Diego Ramos en su Trabajo Fin de Grado [31] (Véase la Ilustración 36)

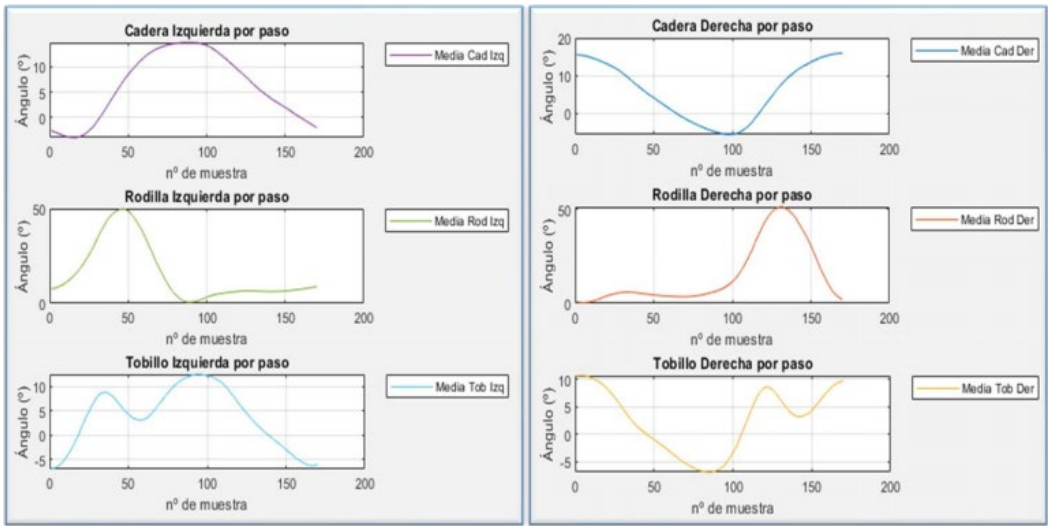

Ilustración 36. Representación del ángulo, en grados, para un paso estándar para todas las articulaciones que componen un exoesqueleto

Los datos de las articulaciones se recogen en un fichero llamado 'Reconstrucción_final.mat' y contiene los ángulos que necesita la ejecución de un paso completo, compuesto por un inicio, un paso, y una finalización hasta el estado de reposo inicial. La representación gráfica de estos datos consiste en el uso de la función 'plot3d()' de MATLAB, la cual permite insertar planos, cuerpos en 3d y elementos dadas unas coordenadas. Nuevamente, el Proyecto que recoge los datos de los ángulos incluye una herramienta para representar gráficamente los mismo a través de una simulación de unas piernas (función 'grafica_angulos_y_cuerpo(angulos,f,frame)'. A este trabajo le aplicaremos una serie de modificaciones para que sea utilizable en nuestro Proyecto:

- Cambio de nombre por 'simulador(angulos,cont,estado)'.
- El plano que simula el suelo adquirirá un color verde cuando reciba, a través del argumento de entrada 'estado' un 2 (marcha) y un color blanco cuando reciba un 1 (paro). De esta manera, será más visible el estado que está adquiriendo el simulador.

```
%% Graficación de los elementos
     %suelo
        x = [-400 -400 400 400];  %Largo del terreno
        y = [-250 250 250 -250];  %Ancho del terreno
        z = [-15 -15 -15 -15];    %Altura del terreno
        if estado==1
            patch(x,y,z,'White')
```

```
        elseif estado==2
            patch(x,y,z,'green')
        end
```

La función ‘simulador(angulos,cont,estado)’ se puede revisar de manera completa en el Anexo V.

El proceso de caminar está compuesto por:

- Aceleración inicial
- Balanceo medio
- Desaceleración final

Una secuencia formada por un paso, estaría compuesta por una aceleración inicial, un balanceo medio y una desaceleración final. Sin embargo, una secuencia formada por indefinidos pasos, estaría formada por una aceleración inicial, una repetición iterativa de balanceos medios y una desaceleración final. El archivo de valores de ángulos que tenemos conforma una secuencia de un paso, por tanto, es necesario diseccionar dicho archivo en tres secciones para poder ejecutar tanto tiempo como el sistema detecte que se solicita caminar el balanceo medio.

Si graficamos los valores de los ángulos en MATLAB, obtenemos lo mostrado en la Ilustración 37:

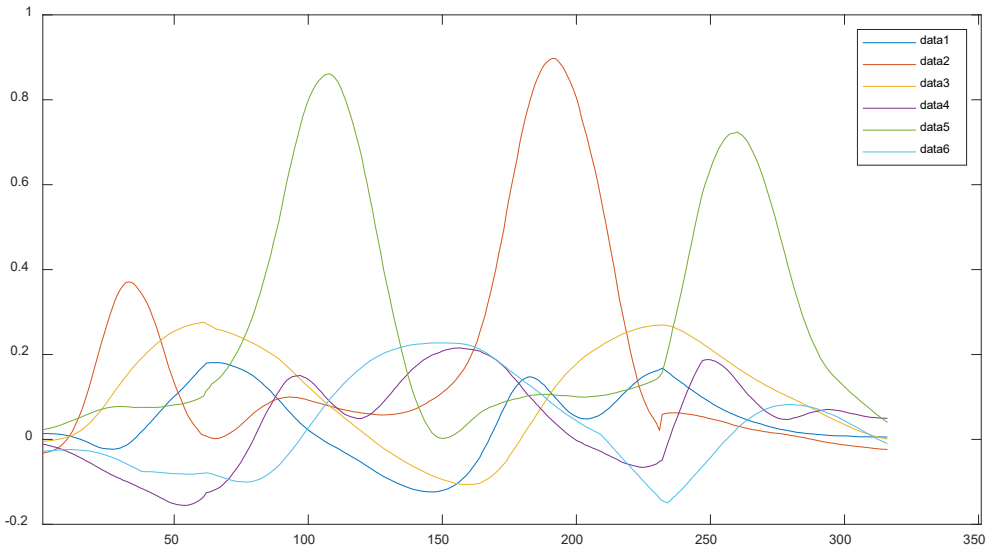

Ilustración 37. Gráfica temporal de los ángulos que adquieren las articulaciones del tren inferior humano en un paso completo. La leyenda equivale, de arriba a abajo, al tobillo, rodilla y cadera de la pierna derecha, y tobillo, rodilla y cadera izquierda, respectivamente.

Dicho vector de ángulos contiene 316 filas de datos, es decir, el paso estándar se ha reconstruido a partir de 316 muestras. De las 316, podemos observar que las correspondientes

a la aceleración inicial son las muestras 1 a 64, mientras que las que corresponden a la desaceleración final son las muestras 221 a 316. Evidentemente, el resto de muestras incluidas desde la 65 a la 220 corresponden al balanceo medio [31]. Supongamos, por tanto, que queremos reproducir el movimiento de un tren inferior ejecutando cuatro pasos. El resultado vendría esquematizado por la Ilustración 38:

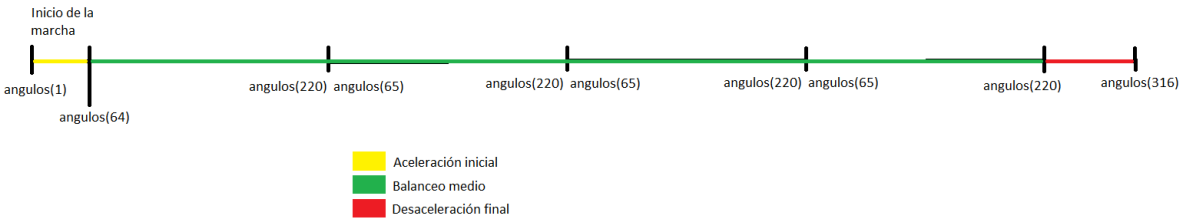

Ilustración 38. Esquema de ejecución de un total de cuatro pasos

5.1. Algoritmo de ejecución del simulador

La función 'simulador(angulos,cont,estado)' es una función que se encarga de graficar todos los elementos del simulador de exoesqueleto, dados unos ángulos en concreto (6 en total). Necesita de unas líneas de código que invoquen a la función con los ángulos correctos en cada momento y de una manera efectiva según lo comentado en los párrafos anteriores, evitando dejar al usuario a mitad de un paso. El Apartado que estamos desarrollando a continuación muestra el trabajo realizado a partir de la herramienta comentada anteriormente, es decir, usaremos dicha herramienta como apoyo gráfico para desarrollar nosotros un algoritmo que le envíe los datos necesarios en cada momento.

Supongamos que el sistema está infiriendo un estado 'paro' de manera indefinida. Lo que debe ocurrir cuando llegue un estado 'marcha' es lo siguiente:

1. Se debe ejecutar la aceleración inicial una vez.
2. Se debe ejecutar el balanceo medio

El sistema ejecutará de manera repetida el balanceo medio, comprendido entre las posiciones 65 y 220 del total de 316 ángulos que tenemos en el archivo, siempre y cuando siga el sistema infiriendo estados de 'marcha' de manera continuada.

Cuando llega por primera vez un estado 'paro', el sistema recordará dicha orden y procederá a finalizar con normalidad el balanceo medio y, cuando llegue al final del balanceo medio, en lugar de repetirlo como ocurría anteriormente, ejecutará una desaceleración final hasta llegar al estado de reposo.

Las variables del simulador serían las siguientes:

```
%Variables para el simulador
load('Angulos.mat');
[N_angulos,articulaciones]=size(Angulos);
cont=1; %Sera el que vaya graficando cada fila de angulos
empezando=0; %Controlará el transitorio de inicio
terminando=0; %Controlará el transitorio de fin
AngulosReposo=Angulos(N_angulos,:);
N_inicio=64; %Filas que corresponden al comienzo
N_fin=95; %Filas que corresponden al final
velocidad=10; %En muestras por 8/128 seg.
```

- N_angulos y articulaciones poseen el número de muestras que tiene el archivo de ángulos (316) y articulaciones (6), respectivamente.
- Cont, inicializado a 1, se encarga de controlar la fila de ángulos que debe ser enviada. Por ejemplo, cuando cont=65, se enviará la primera fila de ángulos correspondiente al balanceo medio.
- Empezando es una variable de control que adquiere el valor '1' cuando se recibe un estado 'marcha'. Volverá a su valor original '0' cuando finalice la ejecución de la aceleración inicial.
- Terminando es otra variable de control que adquiere el valor '1' cuando se recibe un paro mientras se está ejecutando un paso. Se encarga de, una vez llegado al final del balanceo medio, ejecutar la desaceleración final y no repetir el balanceo medio. Volverá a su estado original '0' cuando finalice la desaceleración final.
- AngulosReposo contiene la primera fila de ángulos, correspondiente al estado de reposo del simulador.
- N_inicio y N_fin contienen la cantidad de muestras que componen las secciones de aceleración inicial y desaceleración final, respectivamente.
- Velocidad, en muestras por 8/128 segundos, se encarga de variar la velocidad de ejecución del simulador. Un valor de 10, por ejemplo, indica que el contador va a aumentar de 10 en 10 y no graficará 9 ángulos intermedios.

El controlador del simulador se ejecuta en el bucle del algoritmo de predicción en tiempo real (código completo en Anexo III (*offline*) y Anexo IV (*online*)) que será desarrollado más adelante en el Apartado 6. Las líneas correspondientes a dicho controlador serían las siguientes:

```
%Simulador de exoesqueleto
  %sincronizador para los pasos
  if cont >= N_angulos
      cont=cont-N_angulos+1;
  end
  if n>2 && simulador_on==1
```

```
      if estimacion_final(end)==2
         if (estimacion_final(end-1)==1 || empezando==1) &&
terminando==0;
            if empezando==0
                cont=1;
            end
            empezando=1;
            %empezando
            simulador(Angulos(cont,:),cont,estimacion_final(end));
            Mo(cont)=getframe;
            if cont>=N_inicio
                empezando=0;
            end
         end
         if estimacion_final(end-1)==2 && empezando==0 && terminando ==
0;
            %caminando
            simulador(Angulos(cont,:),cont,estimacion_final(end));
            Mo(cont)=getframe;
            if cont >= N_angulos-N_fin
                cont=N_inicio;
            end
         end
      end
      if estimacion_final(end)==1
         if (estimacion_final(end-1)==2)
             terminando=1;
         end
         if (cont<=N_angulos-N_fin-velocidad && cont >=velocidad) &&
terminando==1
             %caminando
             simulador(Angulos(cont,:),cont,estimacion_final(end));
             Mo(cont)=getframe;
         end
         if (estimacion_final(end-1)==1 && terminando==0)
             %parado
             simulador(AngulosReposo,cont,estimacion_final(end));
             Mo(cont)=getframe;
         end
         if (cont>N_angulos-N_fin-velocidad || cont <velocidad) &&
terminando==1
             %terminando
             simulador(Angulos(cont,:),cont,estimacion_final(end));
             Mo(cont)=getframe;
             if (cont>=N_angulos-velocidad || cont <=velocidad) &&
terminando==1
                 terminando=0;
             end
         end
      end

  end
  cont=cont+velocidad;
```

Dicho código responde a las exigencias planteadas en los párrafos anteriores. En la línea final se encuentra la modificación de 'cont', el cual aumenta según la velocidad prefijada. Es importante que dicho contador retorne al inicio del balanceo medio cuando supere el valor de 220, acción que se ejecuta en el primer 'if' del código. El resto de líneas de código se encargan

de definir los estados en los que se encuentra el sistema, ya sea queriendo iniciar la marcha, caminando, caminando pero queriendo parar o terminando y queriendo parar. Podemos ver, en la Ilustración 39, una serie de *frames* de la ejecución del simulador 3D, donde parte del reposo, comienza a andar y, cuando llega un estado 'paro', finaliza el paso.

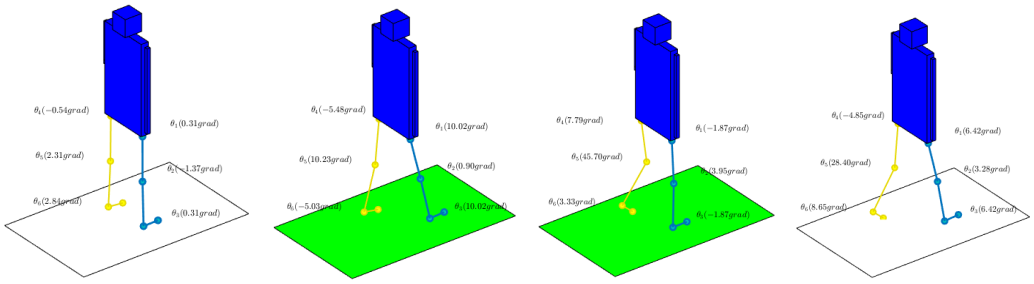

Ilustración 39. De izqueirda a derecha, posibles distintas fases de un paso: Reposo, inicio, repetición y finalización cuando se solicita el paro.

6. Algoritmo de predicción en tiempo real

Hemos visto en el Apartado 4 cómo se pueden inferir los estados del pensamiento del usuario a partir de cuatro ensayos EEG, y usando tres para generar modelos y uno para obtener conclusiones. El objetivo final de este Proyecto es el desarrollo de un sistema que permita a un usuario, usando el equipo Emotiv EPOC, mover un exoesqueleto al mismo tiempo que ejerce un pensamiento. Es, por tanto, necesario desarrollar un algoritmo que realice las etapas desarrolladas en el Apartado 4 en tiempo real.

La idea es recoger los datos que genere el equipo y, cada ciclo de medio segundo, realizar predicciones a partir de la última ventana de 1 segundo de datos. Un esquema del funcionamiento del sistema sería el siguiente Ilustración 40:

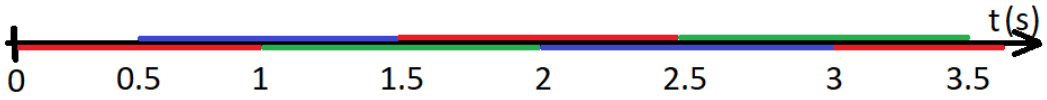

Ilustración 40. Esquema temporal del algoritmo en tiempo real. Cada franja de color representa la ventana de datos que se ha usado para obtener la predicción. De esa manera, se obtienen predicciones cada medio segundo con ventanas de datos de 1 segundo

6.1. Desarrollo de algoritmo de predicción a partir de una sesión grabada: predicción *offline*

Un primer paso para la obtención de un algoritmo que opere en tiempo real sería obtener un algoritmo que opere en tiempo real, pero leyendo los datos del experimento 4. El algoritmo, en lugar de obtener los datos del equipo, leería la última ventana de 1 segundo respecto a la posición temporal del ensayo en la que se encuentre de tal manera que sea lo más fiel a un tiempo real verdadero. De esta manera, lograríamos una primera versión del algoritmo final, al que sólo habría que sustituirle la obtención de datos del 'Exp04.csv' por la obtención de datos invocando librerías de Emotiv.

El sistema se compone de dos algoritmos:

- 'GenerarModelos.m' se encarga de, a partir de 'Exp1.m', 'Exp2.m' y 'Exp3.m' que genera el algoritmo de toma de datos desarrollado en el Apartado 3, obtener los modelos de predicción basados en *Supervised dFasArt.* Algoritmo completo en Anexo II.

- ‘PrediccionOffline.m’ toma los modelos generados en el algoritmo anterior y va recorriendo el vector de datos EEG en bruto extraído de ‘Exp4.csv’, extrayendo ventanas para procesarlas, cotejarlas con los modelos cargados, obtener una estimación y ejecutar el simulador (explicado en detalle en el Apartado 5. Simulador 3D de exoesqueleto). Algoritmo completo en Anexo III.

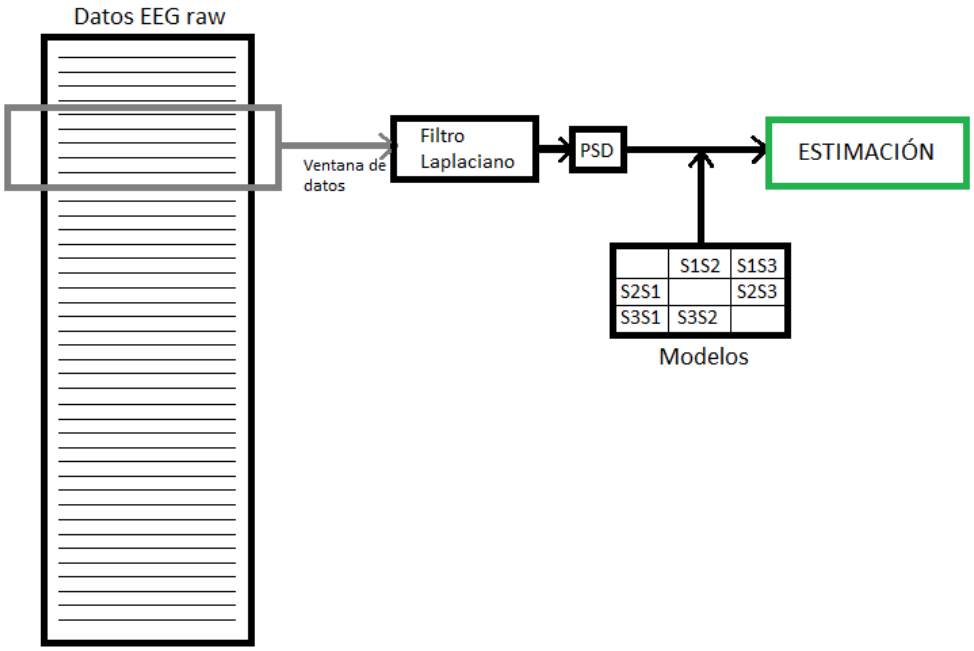

Ilustración 41. Esquema resumen del funcionamiento del algoritmo 'PrediccionOffline.m'

El algoritmo de predicción *offline* es un algoritmo que está pensado para poder ser modificado de la manera más sencilla posible para así generar el algoritmo de predicción *online* que trabaje, en lugar de con los datos grabados en ‘Exp4.csv’, con los datos obtenidos directamente del casco. Por ello, se ha desarrollado conforme al esquema mostrado en la Ilustración 41 de la siguiente manera:

- Se realiza la obtención de los datos y de los modelos.

```
% Lectura de datos
DATOS = csvread('Exp4.csv',1,0);
INPUT = DATOS(:,[2:15])';
[x,y] = ExtractSensorPosition();
[M,N]=size(INPUT);
M=M-2;

(…)

%Carga de modelos
load('Modelos.mat');
for s1=1 : 1 : 3
   for s2=1 : 1 : 3
       if s1 ~= s2
```

```
            Modelo(s1,s2).estado.T(1:end)=0;
        end
    end
end
```

- Se ejecuta un bucle, que inicia a 128 muestras desde el comienzo (para esperar a cumplir la primera ventana de 1 segundo). Dicho bucle se repetirá cada 8 muestras hasta recorrer todo el vector de datos de 'Exp4.csv'. Las partes del bucle que se ejecutarán en cada ciclo son:
    - Laplaciana y PSD de la ventana de datos en cuestión, emulando que se ha recibido una ventana de datos a través del casco y se procesa.

```
%Laplaciana y PSD de la ventana
  [filas,columnas]=size(OUTPUT);
  OUTPUT=[OUTPUT LaplacianFilterZonesSetV(INPUT(:,columnas+1:i),x,y)];
  dato=[];
  for j= 1 : 1 : M
        dato1 = pwelch(OUTPUT(j,i-ventana+1:i) - mean(OUTPUT(j,i-
ventana+1:i)),[],[],F,Fs);
        dato = [dato dato1];
  end
  etiqueta(k) = mode(DATOS(i-ventana+1:i,1));
  lista = 1 : 1 : length(dato);
```

    - Test y predicciones a partir de los datos, de manera similar a como se hacía en el algoritmo 'batch.m' para obtener modelos y estimaciones al mismo tiempo (Véase Apartado 4. Generación de modelos de predicción y estimación de estados del pensamiento)

```
%Test
  for s1=1 : 1 : 3
      for s2=1 : 1 : 3
          if s1 ~= s2
             Modelo(s1,s2) =
predecir_SdFasArt_seleccion(Modelo(s1,s2),dato,lista,Modelo(s1,s2).parame
tros.delta,Modelo(s1,s2).parametros.At);
             estimacion(end+1) = Modelo(s1,s2).salida.prediccion;
          end
      end
  end

  %Predicciones
  if k == desplazamiento
      estimacion_final(n) = mode(estimacion);
      etiqueta_final(n) = mode(etiqueta);
      if estimacion_final(n) ==  etiqueta_final(n)
          aciertos = aciertos + 1;
      end
      k=0;
      n=n+1;
      estimacion=[];
  end
```

- Simulador de exoesqueleto, el cual se detallará en profundidad en el Apartado 5. Dicho simulador utiliza la predicción correspondiente en cada bucle de ejecución y, dependiendo del estado actual en el que se encuentre el simulador, enviará a la función 'simulador(angulos,contador,estimación)' los valores correspondientes. Código completo de la función 'simulador' en Anexo V. Más detalles en el Apartado 5.

La salida de la ejecución del algoritmo 'PrediccionOffline.m' será el simulador 3D, tal y como se muestra en la Ilustración 42.

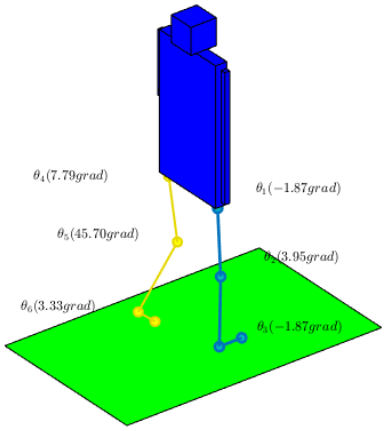

Ilustración 42. Salida que se muestra en la ejecución de 'PrediccionOffline.m', donde se genera una Figura y el simulador comienza a moverse.

6.1.1. Tiempos de ejecución del bucle

Si se realiza una medición del tiempo que tarda en ejecutar todo el bucle el algoritmo 'PrediccionOffline.m', el resultado serán 193.25 segundos. El experimento original 'Exp4.csv' dura un total de 150 segundos, por lo que está habiendo un *offset* de tiempo de 43.25 segundos. Esto sería un inconveniente cuando se aplique el algoritmo de predicción *online*, puesto que significaría que el sistema en su ejecución pierde 43.25 segundos, además del retraso temporal que llevará incluida la interfaz USB de adquisición de datos, produciendo así retrasos y errores no deseados.

Vamos a medir en cada bucle cuánto tiempo lleva ejecutar cada uno de los tres bloques principales que lo componen: Filtrado de señal, obtención de estado y ejecución del simulador.

Para ello, almacenamos en un vector los tiempos que ha tardado en ejecutarse cada bloque durante el bucle, obteniendo los tiempos de la Ilustración 43:

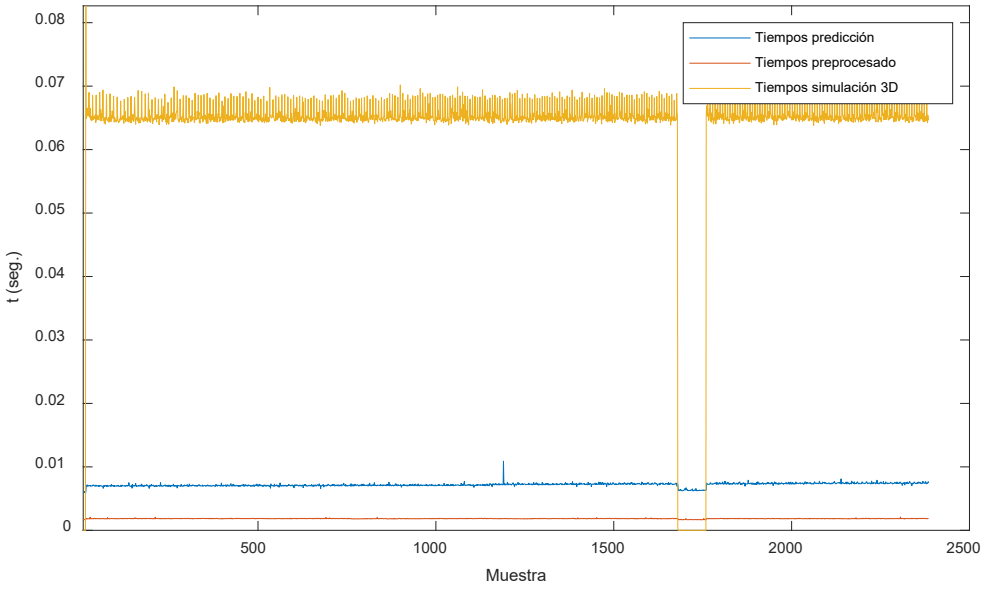

Ilustración 43. Tiempos necesarios para ejecutar, en cada iteración, el filtrado (azul), estimaciones (rojo) y simulación (amarillo)

La conclusión principal es que el proceso de simular al exoesqueleto consume muchos recursos computacionales y, en consecuencia, tiempo. Podemos ver, incluso, picos de tiempo que llegan a los 0.4 segundos y un pico de 0.78 segundos. Esto puede ser debido a que las labores gráficas deberían ser ejecutadas en un PC por la GPU, sin embargo, MATLAB trabaja exclusivamente con la CPU, ralentizando el proceso de graficado. Este problema es inherente a la imposibilidad de usar el exoesqueleto real. Si tuviéramos la posibilidad de usarlo, el proceso se limitaría a enviar los ángulos necesarios para cada instante al controlador del exoesqueleto sin necesidad de procesar ningún elemento gráfico.

6.2. Desarrollo de algoritmo de predicción a partir de datos en tiempo real obtenidos del Emotiv EPOC: Predicción *online*

De manera análoga al Apartado 6.1, podemos obtener un algoritmo que implemente un BCI para que el usuario, usando el equipo Emotiv EPOC, pueda controlar un exoesqueleto. En esta ocasión, debido a las limitaciones que conlleva usar el simulador 3D, se ha propuesto un pseudocódigo modificando el algoritmo de predicción *offline*, como posible Proyecto futuro

cuando se permita el uso del exoesqueleto. El esquema de funcionamiento sería el mostrado en la Ilustración 44:

Ilustración 44. Esquema resumen del funcionamiento del algoritmo 'PrediccionOnline.m'

El pseudocódigo, basado en 'PrediccionOffline.m', consta de las siguientes partes:

- Invocación de librerías de Emotiv EPOC para lectura de datos del equipo
- Bucle 'while true' para que el algoritmo se ejecute de manera indefinida hasta que el usuario lo solicite.

```
tic
while true
    %Nuevo estado cada 10 segundos
    (…)
    %Lectura cada 0.5 segundos
    (…)
end
```

  - Cálculo de nuevo estado cada 10 segundos dentro del bucle.

```
%Nuevo estado cada 10 segundos
    if mod(estimaciones,20)==0
        estados(end+1)=randi([1,2]);
    end
```

  - Lectura de datos, procesado de los mismos, predicción y simulación, cada 0.5 segundos.

```
if toc>=0.5

        %Lectura de datos de equipo
        (…)

        %Laplaciana y PSD de la ventana
        (…)

        %Test y Predicciones
        (…)

        %Simulador de exoesqueleto
        (…)
        tic;
```

```
end
```

El código completo se puede revisar en el Anexo IV.

7. Conclusiones

A lo largo de la literatura se han recogido todos y cada uno de los pasos ejercidos para lograr el objetivo principal propuesto: Lograr un sistema BCI que permita al usuario ser asistido por un exoesqueleto para realizar la labor de caminar, previa colocación del mismo en el tren inferior. La conclusión principal es que el objetivo ha sido cumplido y el Proyecto será completado en cuanto se pueda testar todo el *software* desarrollado en el exoesqueleto real, ya que una de las limitaciones que ha implicado la pandemia por SARS-CoV-2 ha sido la imposibilidad de trabajar en el exoesqueleto real.

Podemos destacar que el Proyecto se ha realizado buscando siempre utilizar equipos EEG asequibles para un usuario medio y programas software compatibles con cualquier PC medio, sean cual sean sus características. Por ello, se ha usado como equipo de captación EEG un Emotiv EPOC v1, *hardware* que se encuentra dentro de la gama de productos que Emotiv ofrece a un precio no superior a 1.000 dólares americanos, siendo este sensiblemente inferior a equipos de mayor calidad.

Además, el uso de una herramienta como MATLAB, con una elevada cantidad de bibliografía asociada, una gran comunidad que le da soporte y siendo una de las soluciones principales que adopta la comunidad científica y tecnológica como herramienta de cálculo matemático en labores docentes, de investigación y de desarrollo, aporta al Proyecto un valor añadido acerca de las posibles implementaciones futuras de mejoras, alternativas y corrección de errores.

Por tanto, vamos a enumerar los *ítems* alcanzados a lo largo de todo el proceso:

1. Se ha modificado el *software* de adquisición de datos EEG, desarrollado por el Dr. Juan Antonio Martínez León en su tesis doctoral [30], adaptando y modificando el mismo a nuestras necesidades particulares.
2. Se ha desarrollado un algoritmo que es capaz de orientar al usuario sobre los datos EEG grabados, obteniendo porcentajes de acierto y matrices de confusión, permitiendo así dotar al usuario de un entrenamiento para determinar las mejores técnicas de cara a la fiabilidad y robustez del sistema.
3. Se han logrado unos resultados, por parte del usuario, aceptables en cuanto a porcentajes de acierto, demostrando la viabilidad y valor del Proyecto.

4. Se ha implementado una alternativa gráfica en sustitución de un exoesqueleto real debido a la imposibilidad de utilización del mismo.

7.1. Líneas futuras

Uno de los valores añadidos de este Proyecto es, sin duda, la posibilidad de desarrollo, mejoría y corrección de errores sobre el mismo, debido a la utilización de *hardware* simple, económico y universal, y *software* sencillo y con una amplia comunidad de usuarios que le da servicio. Además, debido distintas limitaciones existen objetivos planteados que no se han podido completar. Por ello, al realizar este trabajo se han puesto de manifiesto las siguientes áreas de mejoría y posibles continuaciones al Proyecto:

1. Utilización de otro tipo de tecnologías EEG, mirando sobre todo al tipo de sensor utilizado. En el Proyecto se han utilizado sensores húmedos que requieren estar constantemente hidratados para reducir la impedancia de contacto existente entre el mismo y el cuero cabelludo. Esta tecnología, aunque es simple, no es recomendable para una persona que vaya a utilizar este equipo de manera prolongada. Una de las líneas futuras más importantes será valorar la utilización de sensores secos o intracraneales.
2. Estudio de la universalidad del equipo, aplicando el sistema desarrollado en un grupo suficiente de personas que permita obtener conclusiones certeras acerca de la viabilidad del equipo a gran escala. Se sabe que entre un 15 y un 30% de personas se consideran, a efectos de EEG, *analfabetas,* es decir, que métodos clásicos de procesado EEG no van a obtener buenos resultados [32]. Una línea futura sería comprobar si el porcentaje de usuarios fuera de valores óptimos se aproxima a esos porcentajes de analfabetismo EEG.
3. Desarrollo de modelos universales que se adapten a cualquier usuario, evitando así el uso de entrenamientos como forma de obtener modelos personalizados al usuario. DE esta manera estaríamos ante un sistema de tipo *plug-and-play* (Enchufar, conectar y usar).
4. Implementación del sistema en un exoesqueleto real, desarrollando una interfaz de conexión entre los algoritmos de predicción y los algoritmos de control del exoesqueleto y utilizando *software* relacionado al desarrollado en la literatura para implementar un simulador 3D usando los ángulos de un paso estándar.

5. Uso de *software* 3D específico, como el entorno ROS, como simulador 3D del exoesqueleto para paliar las limitaciones gráficas que posee la función 'plot3d()' usada en el Apartado 5, para poder aprovechar de manera más eficiente las ventajas de los algoritmos de predicción *online* los algoritmos de predicción *offline* y *online* desarrolladas en el Apartado 6.

7.2. Evaluación de competencias desarrolladas

Un Trabajo Fin de Grado tiene, entre sus objetivos, permitir al estudiante desarrollar competencias y habilidades que haya adquirido durante el transcurso de su etapa de formación universitaria. Por ello, las competencias que se han desarrollado al realizad este Proyecto han sido:

- Demostración de conocimientos en un área concreta de estudio, apoyada en unos conocimientos básicos adquiridos durante la formación universitaria e incrementada a través del proceso de investigación mediante bibliografías complementarias.
- Capacidad de reunir e interpretar datos relevantes para emitir juicios que incluyan reflexiones en temas relevantes de índole científica, social o ética.
- Transmisión de conocimientos e información tanto a un público especializado como no especializado.
- Desarrollo de la capacidad de redacción, firma y desarrollo de proyectos en el ámbito de la Ingeniería Industrial.
- Capacidad de resolver problemas con iniciativa, creatividad y razonamiento crítico.
- Utilización de manera solvente de los recursos de información disponibles.

Referencias

[1] A. Alfageme, «La Ley de Dependencia prevé que los tetrapléjicos tengan un asistente personal,» *El País,* 6 Enero 2006.

[2] P. Manns y K. Chad, «Components of Quality of Life for Persons With a Quadriplegic and Paraplegic Spinal Cord Injury,» *Qualitative Healt Research,* vol. 11, nº 6, pp. 795-811, 2001.

[3] M. Eidel y A. Kübler, «Wheelchair Control in a Virtual Environment by Healthy Participants Using a P300-BCI Based on Tactile Stimulation: Training Effects and Usability,» *Frontiers in Human Neuroscience,* vol. 14, p. 265, 2020.

[4] A. Cruz, G. Pires, A. Lopes, C. Carona y U. Nunes, «A Self-Paced BCI With a Collaborative Controller for Highly Reliable Wheelchair Driving: Experimental Tests With Physically Disabled Individuals,» *IEEE Transactions on Human-Machine Systems,* vol. 51, nº 2, pp. 109-119, 2021.

[5] F. Wang, Z. Xu, W. Zhang, S. Wu, Y. Zhang y S. Coleman, «An Adaptive Control Approach for Intelligent Wheelchair Based on BCI Combining with QoO,» de *2020 International Joint Conference on Neural Networks (IJCNN)*, Glasgow, U.K., 2020.

[6] J. Chen, C. Wu, Y. Lin, Y. Kuo y C. Kuo, «Mechatronic Implementation and Trajectory Tracking Validation of a BCI-based Human-wheelchair Interface,» de *8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, New York (NY), USA, 2020.

[7] A. Manvi, A. Masood y K. Mohanchandra, «Brain Operated Wheelchair Using a Single Electrode EEG Device and BCI,» *International Journal of Artificial Intelligence,* vol. 7, nº 1, pp. 1-6, 2020.

[8] J. Blasco, N. Pavón y J. Feliú, «Sistema de control de un Exoesqueleto robótico usando ROS,» de *TFE UPCT*, Cartagena, 2019.

[9] N. Pavón, J. López y J. Feliu, «IoT Architecture for Smart Control of an Exoskeleton Robot in Rehabilitation by Using a Natural User Interface Based on Gestures,» *Journal of Medical Systems,* vol. 44, pp. 1-10, 2020.

[10] M. Almonacid, «Voting Strategy to Enhance Multimodel EEG-BasedClassifier Systems for Motor Imagery BCI,» *IEEE system journal,* vol. 10, pp. 1082-1089, 2016.

[11] L. Lledo, F. Badesa, M. Almonacid, J. Cano, J. Sabater-Navarro, E. Fernández y N. García-Aracil, «Supervised and Dynamic Neuro-Fuzzy Systems to Classify Physiological Responses in Robot-Assisted Neurorehabilitation,» *PLOS One,* vol. 10, nº 5, pp. 1-16, 2015.

[12] J. Martínez, J. Cano y J. Ibarrola, «Feature Selection Applying Statistical and Neurofuzzy Methods to EEG-Based BCI,» *Computational Intelligence and Neuroscience,* vol. 2015, pp. 1-17, 2015.

[13] J. Cano, J. Ibarrola y M. Almonacid, «Improving Motor Imagery Classification with a new BCI Design using Neuro-Fuzzy S-dFasArt Architecture,» *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* vol. 10, nº 1, pp. 2-7, 2013.

[14] M. Hämäläinen, R. Hari, R. J. Ilmoniemi, J. Knuutila y O. V. Lounasmaa, «Magnetoencephalography - theory, instrumentation, and applications to noninvasive studies of the working human brain,» *Reviews of Modern Physics,* vol. 65, nº 2, pp. 413-497, 1993.

[15] C. Noback, N. Strominger, R. Demarest y D. Ruggiero, «The human nervous system: structure and function.,» *Springer Science & Business Media,* nº 744, 2005.

[16] R. Douglas Fields, «White Matter Matters,» *Scientific American,* nº 298(3), pp. 54-61, 2008.

[17] H. Jasper y H. Andrews, «Electro-encephalography III. Normal differentiation of occipital and precentral regions in man,» *Archives of Neurology and Psychiatry,* vol. 39, pp. 163-174, 1938.

[18] N. R. Carlson y M. A. Birkett, Fisiología de la conducta, Pearson.

[19] J. Wolpaw, B. Allison, E. Donchin, O. do Nascimento y W. Heetderks, «Workshop on signals andrecording methods,» *IEEE Transactions on neural systems and rehabilitation,* vol. 14, nº 2, pp. 138-141, 2006.

[20] A. Dietrich y R. Kanso, «A review of EEG, ERP and neuroimaging studies of creativity and insight,» *Psychological Bulletin,* vol. 136, nº 5, pp. 822-848, 2010.

[21] S. Smith, «EEG in the diagnosis, classification, and management of patients with epilepsy,» *Journal of Neurology, Neurosurgery & Psychiatry,* vol. 76, nº 2, pp. ii2-ii7, 2005.

[22] P. Kaplan y A. Rossetti, «EEG Patterns and Imaging Correlations in Encephalopathy,» *Journal of Clinical Neurophysiology,* vol. 28, nº 3, pp. 233-251, 2011.

[23] «American Electroencephalographic Society Guidelines for Standard Electrode Position Nomenclature,» *Journal of Clinical Neurophysiology,* vol. 8, nº 2, pp. 200-202, 1991.

[24] P. Nunez, R. Srinivasan, A. Westdorp, R. Wijesinghe y D. Tucker, «"EEG coherency I: Statistics, reference electrode, volume conduction, Laplacians, cortical imaging, and interpretation at,» *Electroencephalogr. Clin. Neurophysiol.,* vol. 103, nº 5, pp. 499-515, 1997.

[25] S. Syam, H. Lakany, R. Ahmad y B. Conway, «Comparing Common Average Referencing to Laplacian Referencing in Detecting Imagination and Intention of Movement for Brain Computer Interface,» *MATEC Web of Conferences,* vol. 140, nº 01028, 2017.

[26] M. Núñez-Peña, M. Corral y C. Escera, «Potenciales evocados cerebrales en el contexto de la investigación psicológica: una actualización,» Barcelona, 2004.

[27] R. Johnson, «For Distinguished Early Career Contribution to Psychophysiology: Award Address,» *Psychophysiology,* vol. 23, nº 4, pp. 367-384, 1985.

[28] E. Garcia y G. Gentiletti, «Interfaz cerebro computadora (ICC) basada en el potencial relacionado con eventos P300: análisis del efecto de la dimensión de la matriz de estimulación sobre su desempeño,» *Revista de ingeniería biomédica,* vol. 2, nº 4, pp. 26-33, 2008.

[29] B. Graimann, G. Pfurtscheller y B. Allison, Brain-Computer Interfaces, Graz, Austria: Springer, 2010.

[30] J. Martínez León, J. Ibarrola Lacalle y J. Cano Izquierdo, «Sistemas de Interfaz Cerebro-Ordenador Basados en Dispositivos EEG de Bajo Coste y Modelos Neurodifusos Aplicados a la Imaginación de Movimiento,» de *Tesis Doctoral UPCT*, Cartagena, 2018.

[31] D. Ramos, J. Ibarrola y J. Cano, «Identificación de patrones de marcha mediante un análisis de movimiento para su aplicación a un exoesqueleto,» de *TFE UPCT*, Cartagena, 2020.

[32] T. Dickhaus, C. Sannelli, C. Müller, G. Curio y B. Blankertz, «Predicting BCI performance to study BCI illiteracy,» *BMC Neuroscience,* vol. 10, nº 1, p. 84, 2009.

Anexos

Anexo I. Algoritmo de obtención de datos EEG modificado

```
function eeglogger_statusesBCIForms(ExperimentNumber, handles)

experiment_number = str2double(ExperimentNumber);
set(handles.PredictionWord,'String',' ');

structs.InputSensorDescriptor_struct.members=struct('channelId',
'EE_InputChannels_enum', 'fExists', 'int32', 'pszLabel', 'cstring',
'xLoc', 'double', 'yLoc', 'double', 'zLoc', 'double');
enuminfo.EE_DataChannels_enum=struct('ED_COUNTER',0,'ED_INTERPOLATED',1,'
ED_RAW_CQ',2,'ED_AF3',3,'ED_F7',4,'ED_F3',5,'ED_FC5',6,'ED_T7',7,'ED_P7',
8,'ED_O1',9,'ED_O2',10,'ED_P8',11,'ED_T8',12,'ED_FC6',13,'ED_F4',14,'ED_F
8',15,'ED_AF4',16,'ED_GYROX',17,'ED_GYROY',18,'ED_TIMESTAMP',19,'ED_ES_TI
MESTAMP',20,'ED_FUNC_ID',21,'ED_FUNC_VALUE',22,'ED_MARKER',23,'ED_SYNC_SI
GNAL',24);
enuminfo.EE_CognitivTrainingControl_enum=struct('COG_NONE',0,'COG_START',
1,'COG_ACCEPT',2,'COG_REJECT',3,'COG_ERASE',4,'COG_RESET',5);
enuminfo.EE_ExpressivAlgo_enum=struct('EXP_NEUTRAL',1,'EXP_BLINK',2,'EXP_
WINK_LEFT',4,'EXP_WINK_RIGHT',8,'EXP_HORIEYE',16,'EXP_EYEBROW',32,'EXP_FU
RROW',64,'EXP_SMILE',128,'EXP_CLENCH',256,'EXP_LAUGH',512,'EXP_SMIRK_LEFT
',1024,'EXP_SMIRK_RIGHT',2048);
enuminfo.EE_ExpressivTrainingControl_enum=struct('EXP_NONE',0,'EXP_START'
,1,'EXP_ACCEPT',2,'EXP_REJECT',3,'EXP_ERASE',4,'EXP_RESET',5);
enuminfo.EE_ExpressivThreshold_enum=struct('EXP_SENSITIVITY',0);
enuminfo.EE_CognitivEvent_enum=struct('EE_CognitivNoEvent',0,'EE_Cognitiv
TrainingStarted',1,'EE_CognitivTrainingSucceeded',2,'EE_CognitivTrainingF
ailed',3,'EE_CognitivTrainingCompleted',4,'EE_CognitivTrainingDataErased'
,5,'EE_CognitivTrainingRejected',6,'EE_CognitivTrainingReset',7,'EE_Cogni
tivAutoSamplingNeutralCompleted',8,'EE_CognitivSignatureUpdated',9);
enuminfo.EE_EmotivSuite_enum=struct('EE_EXPRESSIV',0,'EE_AFFECTIV',1,'EE_
COGNITIV',2);
enuminfo.EE_ExpressivEvent_enum=struct('EE_ExpressivNoEvent',0,'EE_Expres
sivTrainingStarted',1,'EE_ExpressivTrainingSucceeded',2,'EE_ExpressivTrai
ningFailed',3,'EE_ExpressivTrainingCompleted',4,'EE_ExpressivTrainingData
Erased',5,'EE_ExpressivTrainingRejected',6,'EE_ExpressivTrainingReset',7)
;
enuminfo.EE_CognitivAction_enum=struct('COG_NEUTRAL',1,'COG_PUSH',2,'COG_
PULL',4,'COG_LIFT',8,'COG_DROP',16,'COG_LEFT',32,'COG_RIGHT',64,'COG_ROTA
TE_LEFT',128,'COG_ROTATE_RIGHT',256,'COG_ROTATE_CLOCKWISE',512,'COG_ROTAT
E_COUNTER_CLOCKWISE',1024,'COG_ROTATE_FORWARDS',2048,'COG_ROTATE_REVERSE'
,4096,'COG_DISAPPEAR',8192);
enuminfo.EE_InputChannels_enum=struct('EE_CHAN_CMS',0,'EE_CHAN_DRL',1,'EE
_CHAN_FP1',2,'EE_CHAN_AF3',3,'EE_CHAN_F7',4,'EE_CHAN_F3',5,'EE_CHAN_FC5',
6,'EE_CHAN_T7',7,'EE_CHAN_P7',8,'EE_CHAN_O1',9,'EE_CHAN_O2',10,'EE_CHAN_P
8',11,'EE_CHAN_T8',12,'EE_CHAN_FC6',13,'EE_CHAN_F4',14,'EE_CHAN_F8',15,'E
E_CHAN_AF4',16,'EE_CHAN_FP2',17);
enuminfo.EE_ExpressivSignature_enum=struct('EXP_SIG_UNIVERSAL',0,'EXP_SIG
_TRAINED',1);
enuminfo.EE_Event_enum=struct('EE_UnknownEvent',0,'EE_EmulatorError',1,'E
E_ReservedEvent',2,'EE_UserAdded',16,'EE_UserRemoved',32,'EE_EmoStateUpda
ted',64,'EE_ProfileEvent',128,'EE_CognitivEvent',256,'EE_ExpressivEvent',
512,'EE_InternalStateChanged',1024,'EE_AllEvent',2032);
enuminfo.EE_AffectivAlgo_enum=struct('AFF_EXCITEMENT',1,'AFF_MEDITATION',
2,'AFF_FRUSTRATION',4,'AFF_ENGAGEMENT_BOREDOM',8);
```

```
DataChannels = enuminfo.EE_DataChannels_enum;
DataChannelsNames =
{'ED_COUNTER','ED_INTERPOLATED','ED_RAW_CQ','ED_AF3','ED_F7','ED_F3','ED_
FC5','ED_T7','ED_P7','ED_O1','ED_O2','ED_P8','ED_T8','ED_FC6','ED_F4','ED
_F8','ED_AF4','ED_GYROX','ED_GYROY','ED_TIMESTAMP','ED_ES_TIMESTAMP','ED_
FUNC_ID','ED_FUNC_VALUE','ED_MARKER','ED_SYNC_SIGNAL'};

rectime = 1;
acqtime = 160;

sampFreq = 128;
status_time = 15;
counter_status = 0;
total_samples = (acqtime-10)*sampFreq;
status_samples = status_time*sampFreq;
samples_taken = 0;

status_set_matrix = [
    1 2 1 1 2 1 2 2 1 2 2 1 1 2 1 2 1 2 1 1 2 1 2 2 1 2 2 1 1 2 1 2 1 2 1
1 2 1 2 2 1 2 2 1 1 2 1 2;
    2 1 2 1 1 2 2 1 2 1 1 2 1 2 2 1 2 1 2 1 1 2 2 1 2 1 1 2 1 2 2 1 2 1 2
1 1 2 2 1 2 1 1 2 1 2 2 1;
    2 2 1 2 2 1 1 2 1 2 1 1 2 1 2 1 2 2 1 2 2 1 1 2 1 2 1 1 2 1 2 1 2 2 1
2 2 1 1 2 1 2 1 1 2 1 2 1;
    1 1 2 1 2 1 1 2 2 1 2 1 2 2 1 2 1 1 2 1 2 1 1 2 2 1 2 1 2 2 1 2 1 1 2
1 2 1 1 2 2 1 2 1 2 2 1 2;
    2 1 2 1 1 2 1 2 1 2 2 1 1 2 1 2 2 1 2 1 1 2 1 2 1 2 2 1 1 2 1 2 2 1 2
1 1 2 1 2 1 2 2 1 1 2 1 2];

samplesSinceLastWindow = 0;
output_matrix = zeros(acqtime*sampFreq,length(DataChannelsNames)+1);

if ~libisloaded('edk')
    [nf, w] = loadlibrary('edk','edk','addheader', 'EmoStateDLL',
'addheader', 'edkErrorCode');
else
    disp(['EDK library already loaded']);
end

default = int8(['Emotiv Systems-5' 0]);
AllOK = calllib('edk','EE_EngineConnect','Emotiv Systems-5');

hData = calllib('edk','EE_DataCreate');
calllib('edk','EE_DataSetBufferSizeInSec',1);
eEvent = calllib('edk','EE_EmoEngineEventCreate');
readytocollect = false;
cnt = 0;

%Grabación del experimento
disp('Get ready to record your session')
tic
muestras_tomadas=[];
while(samples_taken < total_samples)
    tiempo=toc;

    %Cambio de estado cada 15 seg.
    if samples_taken >= counter_status*status_samples
        status = status_set_matrix(experiment_number,counter_status+1);
        handles = PresentStatus(status,counter_status,handles);
```

```
        counter_status = counter_status + 1;
    end

    %Obtener muestras
    state = calllib('edk','EE_EngineGetNextEvent',eEvent);
    eventType = calllib('edk','EE_EmoEngineEventGetType',eEvent);
    userID=libpointer('uint32Ptr',0);
    calllib('edk','EE_EmoEngineEventGetUserId',eEvent, userID);
    if strcmp(eventType,'EE_UserAdded') == true
        User_added = 1;
        userID_value = get(userID,'value');
        calllib('edk','EE_DataAcquisitionEnable',userID_value,true);
        readytocollect = true;
    end
    if (readytocollect)
        calllib('edk','EE_DataUpdateHandle', 0, hData);
        nSamples = libpointer('uint32Ptr',0);
        calllib('edk','EE_DataGetNumberOfSample',hData,nSamples);
        nSamplesTaken = get(nSamples,'value');
        samples_taken = samples_taken + nSamplesTaken;
        samplesSinceLastWindow = samplesSinceLastWindow + nSamplesTaken;
        if (nSamplesTaken ~= 0)
            data = libpointer('doublePtr',zeros(1,nSamplesTaken));
            for i = 1:length(fieldnames(enuminfo.EE_DataChannels_enum))
                calllib('edk','EE_DataGet',hData,
DataChannels.([DataChannelsNames{i}]), data, uint32(nSamplesTaken));
                data_value = get(data,'value');
                output_matrix(cnt+1:cnt+length(data_value),i) =
data_value;
            end
            output_matrix(cnt+1:cnt+length(data_value),i+1) = status;

            %Tratamiento de tiempos
            if cnt==0
                escalon=tiempo/length(data_value);
                for j=1:length(data_value)
                    output_matrix(cnt+j,i+2)=(j-1)*escalon;
                end
            else
                tiempo_anterior=output_matrix(cnt,i+2);
                escalon=(tiempo-tiempo_anterior)/length(data_value);
                for j=1:length(data_value)
                    output_matrix(cnt+j,i+2)=tiempo_anterior+j*escalon;
                end
            end
            cnt = cnt + length(data_value);
        end
    end
end

sampRateOutPtr = libpointer('uint32Ptr',0);
calllib('edk','EE_DataGetSamplingRate',0,sampRateOutPtr);
calllib('edk','EE_DataFree',hData);
end_time = find(output_matrix(:,20)==0,1) - 1;

%Guardado de archivos
disp('Finishing your session - creating output file');

DataChannelsNamesExported =
{'STATUS','ED_AF3','ED_F7','ED_F3','ED_FC5','ED_T7','ED_P7','ED_O1','ED_O
```

```
2','ED_P8','ED_T8','ED_FC6','ED_F4','ED_F8','ED_AF4','ED_ES_TIMESTAMP','T
iemposNuevos'};
matrix_export = output_matrix(1:end_time,[26 4:17 27]);
fname = strcat(strcat('Exp',num2str(experiment_number)),'.csv');
cell2csv(fname,DataChannelsNamesExported,',');
dlmwrite(fname, matrix_export,'-append');

set(handles.PredictionWord,'String','Finalizado');
handles = PresentStatus(0,counter_status,handles);
```

Anexo II. Algoritmo de generación de modelos

```
clc; clear;
poda=1;
Ar=0.002;

% Preprocesado
for Sesion = 1 : 1 : 3
    file_in=sprintf('Exp%d.csv',Sesion)
    file_out=sprintf('U1S%d.mat',Sesion)
    preprocesado(file_in,file_out);
end

% Aprendizaje
for sa = 1 : 1 : 3
    file_in=sprintf('U1S%d.mat',sa)
    Modelo(sa,sa) = crear_SdFasArt;
    Modelo(sa,sa) = generar_modelo(Modelo(sa,sa),file_in,Ar);
    for sb = 1 : 1 : 3
        if sa ~= sb
            Modelo(sa,sb) = crear_SdFasArt;
        end
    end
end

% Optimización de Parametros
At = zeros(3,3);
delta = zeros(3,3);
for Sesion_1 = 1 : 1 : 3
    for Sesion_2 = 1 : 1 : 3
        if Sesion_1 ~= Sesion_2
           file_in = sprintf('U1S%d.mat',Sesion_2)
           [At(Sesion_1,Sesion_2),delta(Sesion_1,Sesion_2)] =
buscar_parametros(Modelo(Sesion_1,Sesion_1),file_in)
           Modelo(Sesion_1,Sesion_2).parametros.delta =
delta(Sesion_1,Sesion_2);
           Modelo(Sesion_1,Sesion_2).parametros.At =
At(Sesion_1,Sesion_2);
        end
    end
end

% Poda
% sa Aprendizaje
% sb Optimizar parametros & Poda
for sa = 1 : 1 : 3
    for sb = 1 : 1 : 3
        if sa ~= sb
            Modelo(sa,sb) =
Calcular_Modelo_Poda(4,Modelo(sa,sa),sb,delta(sa,sb),At(sa,sb),poda);
        end
    end
end

%Guardado de modelos
save('Modelos','Modelo');
```

Anexo III. Algoritmo de predicción *offline*

```
clc;clear;

% Lectura de datos
DATOS = csvread('Exp4.csv',1,0);
INPUT = DATOS(:,[2:15])';
[x,y] = ExtractSensorPosition();
[M,N]=size(INPUT);
M=M-2;

%Variables filtros
F = (8 : 2 : 30);
Fs = 128;
ventana = 128;
desplazamiento = 8;

%Carga de modelos
load('Modelos.mat');
for s1=1 : 1 : 3
   for s2=1 : 1 : 3
       if s1 ~= s2
           Modelo(s1,s2).estado.T(1:end)=0;
       end
   end
end
aciertos=0;
k=1;
n=1;
estimacion=[];
tiempos=[];

%Variables para el simulador
load('Angulos.mat');
[N_angulos,articulaciones]=size(Angulos);
cont=1; %Sera el que vaya graficando cada fila de angulos
empezando=0; %Controlará el transitorio de inicio
terminando=0; %Controlará el transitorio de fin
AngulosReposo=Angulos(N_angulos,:);
N_inicio=64; %Filas que corresponden al comienzo
N_fin=95; %Filas que corresponden al final
simulador(AngulosReposo,1,1);
saltos=100;

OUTPUT=[]; %Vector donde se pondrán las Laplacianas en cada bucle

fprintf('\n\nPreparado para obtener predicciones\n\n');
for i=ventana : desplazamiento  : N
  %Laplaciana de la ventana
  [filas,columnas]=size(OUTPUT);
  OUTPUT=[OUTPUT LaplacianFilterZonesSetV(INPUT(:,columnas+1:i),x,y)];
  dato=[];

  %PSD
  for j= 1 : 1 : M
        dato1 = pwelch(OUTPUT(j,i-ventana+1:i) - mean(OUTPUT(j,i-
ventana+1:i)),[],[],F,Fs);
        dato = [dato dato1];
  end
```

```
  etiqueta(k) = mode(DATOS(i-ventana+1:i,1)); %posibilidad de ajustar
esto
  lista = 1 : 1 : length(dato);

  %Test
  for s1=1 : 1 : 3
      for s2=1 : 1 : 3
          if s1 ~= s2
             Modelo(s1,s2) =
predecir_SdFasArt_seleccion(Modelo(s1,s2),dato,lista,Modelo(s1,s2).parame
tros.delta,Modelo(s1,s2).parametros.At);
             estimacion(end+1) = Modelo(s1,s2).salida.prediccion;
          end
      end
  end

  %Predicciones
  if k == desplazamiento
      estimacion_final(n) = mode(estimacion);
      etiqueta_final(n) = mode(etiqueta);
      if estimacion_final(n) ==  etiqueta_final(n)
          aciertos = aciertos + 1;
      end
      k=0;
      n=n+1;
      estimacion=[];
  end

  %Simulador de exoesqueleto
  %sincronizador para los pasos
  if cont >= N_angulos
      cont=cont-N_angulos+1;
  end
  if n>2
      if estimacion_final(end)==2
         if (estimacion_final(end-1)==1 || empezando==1) &&
terminando==0;
            if empezando==0
                cont=1;
            end
            empezando=1;
            %empezando
            simulador(Angulos(cont,:),cont,estimacion_final(end));
            Mo(cont)=getframe;
            if cont>=N_inicio
                empezando=0;
            end
         end
         if estimacion_final(end-1)==2 && empezando==0 && terminando ==
0;
            %caminando
            simulador(Angulos(cont,:),cont,estimacion_final(end));
            Mo(cont)=getframe;
            if cont >= N_angulos-N_fin
                cont=N_inicio;
            end
         end
      end
      if estimacion_final(end)==1
         if (estimacion_final(end-1)==2)
             terminando=1;
```

```
          end
          if (cont<=N_angulos-N_fin-saltos && cont >=saltos) &&
terminando==1
              %caminando
              simulador(Angulos(cont,:),cont,estimacion_final(end));
              Mo(cont)=getframe;
          end
          if (estimacion_final(end-1)==1 && terminando==0)
              %parado
              simulador(AngulosReposo,cont,estimacion_final(end));
              Mo(cont)=getframe;
          end
          if (cont>N_angulos-N_fin-saltos || cont <saltos) &&
terminando==1
              %terminando
              simulador(Angulos(cont,:),cont,estimacion_final(end));
              Mo(cont)=getframe;
              if (cont>=N_angulos-saltos || cont <=saltos) &&
terminando==1
                  terminando=0;
              end
          end
      end
  end
  k=k+1;
  cont=cont+saltos;
  tiempos_simulador=[tiempos_simulador toc];
end
aciertos/length(estimacion_final)

plot(etiqueta_final);
hold on;
plot(estimacion_final,'*');
```

Anexo IV. Algoritmo de predicción *online*

```
clc;clear;
simulador_on=1; %1 para encendido

%Librerías para adquisición de datos
structs.InputSensorDescriptor_struct.members=struct('channelId',
'EE_InputChannels_enum', 'fExists', 'int32', 'pszLabel', 'cstring',
'xLoc', 'double', 'yLoc', 'double', 'zLoc', 'double');
enuminfo.EE_DataChannels_enum=struct('ED_COUNTER',0,'ED_INTERPOLATED',1,'
ED_RAW_CQ',2,'ED_AF3',3,'ED_F7',4,'ED_F3',5,'ED_FC5',6,'ED_T7',7,'ED_P7',
8,'ED_O1',9,'ED_O2',10,'ED_P8',11,'ED_T8',12,'ED_FC6',13,'ED_F4',14,'ED_F
8',15,'ED_AF4',16,'ED_GYROX',17,'ED_GYROY',18,'ED_TIMESTAMP',19,'ED_ES_TI
MESTAMP',20,'ED_FUNC_ID',21,'ED_FUNC_VALUE',22,'ED_MARKER',23,'ED_SYNC_SI
GNAL',24);
enuminfo.EE_CognitivTrainingControl_enum=struct('COG_NONE',0,'COG_START',
1,'COG_ACCEPT',2,'COG_REJECT',3,'COG_ERASE',4,'COG_RESET',5);
enuminfo.EE_ExpressivAlgo_enum=struct('EXP_NEUTRAL',1,'EXP_BLINK',2,'EXP_
WINK_LEFT',4,'EXP_WINK_RIGHT',8,'EXP_HORIEYE',16,'EXP_EYEBROW',32,'EXP_FU
RROW',64,'EXP_SMILE',128,'EXP_CLENCH',256,'EXP_LAUGH',512,'EXP_SMIRK_LEFT
',1024,'EXP_SMIRK_RIGHT',2048);
enuminfo.EE_ExpressivTrainingControl_enum=struct('EXP_NONE',0,'EXP_START'
,1,'EXP_ACCEPT',2,'EXP_REJECT',3,'EXP_ERASE',4,'EXP_RESET',5);
enuminfo.EE_ExpressivThreshold_enum=struct('EXP_SENSITIVITY',0);
enuminfo.EE_CognitivEvent_enum=struct('EE_CognitivNoEvent',0,'EE_Cognitiv
TrainingStarted',1,'EE_CognitivTrainingSucceeded',2,'EE_CognitivTrainingF
ailed',3,'EE_CognitivTrainingCompleted',4,'EE_CognitivTrainingDataErased'
,5,'EE_CognitivTrainingRejected',6,'EE_CognitivTrainingReset',7,'EE_Cogni
tivAutoSamplingNeutralCompleted',8,'EE_CognitivSignatureUpdated',9);
enuminfo.EE_EmotivSuite_enum=struct('EE_EXPRESSIV',0,'EE_AFFECTIV',1,'EE_
COGNITIV',2);
enuminfo.EE_ExpressivEvent_enum=struct('EE_ExpressivNoEvent',0,'EE_Expres
sivTrainingStarted',1,'EE_ExpressivTrainingSucceeded',2,'EE_ExpressivTrai
ningFailed',3,'EE_ExpressivTrainingCompleted',4,'EE_ExpressivTrainingData
Erased',5,'EE_ExpressivTrainingRejected',6,'EE_ExpressivTrainingReset',7)
;
enuminfo.EE_CognitivAction_enum=struct('COG_NEUTRAL',1,'COG_PUSH',2,'COG_
PULL',4,'COG_LIFT',8,'COG_DROP',16,'COG_LEFT',32,'COG_RIGHT',64,'COG_ROTA
TE_LEFT',128,'COG_ROTATE_RIGHT',256,'COG_ROTATE_CLOCKWISE',512,'COG_ROTAT
E_COUNTER_CLOCKWISE',1024,'COG_ROTATE_FORWARDS',2048,'COG_ROTATE_REVERSE'
,4096,'COG_DISAPPEAR',8192);
enuminfo.EE_InputChannels_enum=struct('EE_CHAN_CMS',0,'EE_CHAN_DRL',1,'EE
_CHAN_FP1',2,'EE_CHAN_AF3',3,'EE_CHAN_F7',4,'EE_CHAN_F3',5,'EE_CHAN_FC5',
6,'EE_CHAN_T7',7,'EE_CHAN_P7',8,'EE_CHAN_O1',9,'EE_CHAN_O2',10,'EE_CHAN_P
8',11,'EE_CHAN_T8',12,'EE_CHAN_FC6',13,'EE_CHAN_F4',14,'EE_CHAN_F8',15,'E
E_CHAN_AF4',16,'EE_CHAN_FP2',17);
enuminfo.EE_ExpressivSignature_enum=struct('EXP_SIG_UNIVERSAL',0,'EXP_SIG
_TRAINED',1);
enuminfo.EE_Event_enum=struct('EE_UnknownEvent',0,'EE_EmulatorError',1,'E
E_ReservedEvent',2,'EE_UserAdded',16,'EE_UserRemoved',32,'EE_EmoStateUpda
ted',64,'EE_ProfileEvent',128,'EE_CognitivEvent',256,'EE_ExpressivEvent',
512,'EE_InternalStateChanged',1024,'EE_AllEvent',2032);
enuminfo.EE_AffectivAlgo_enum=struct('AFF_EXCITEMENT',1,'AFF_MEDITATION',
2,'AFF_FRUSTRATION',4,'AFF_ENGAGEMENT_BOREDOM',8);
DataChannels = enuminfo.EE_DataChannels_enum;
DataChannelsNames =
{'ED_COUNTER','ED_INTERPOLATED','ED_RAW_CQ','ED_AF3','ED_F7','ED_F3','ED_
FC5','ED_T7','ED_P7','ED_O1','ED_O2','ED_P8','ED_T8','ED_FC6','ED_F4','ED
_F8','ED_AF4','ED_GYROX','ED_GYROY','ED_TIMESTAMP','ED_ES_TIMESTAMP','ED_
FUNC_ID','ED_FUNC_VALUE','ED_MARKER','ED_SYNC_SIGNAL'};
```

```
%Variables filtros
F = (8 : 2 : 30);
Fs = 128;
ventana = 128;
desplazamiento = 8;

%Carga de modelos
load('Modelos.mat');
for s1=1 : 1 : 3
    for s2=1 : 1 : 3
        if s1 ~= s2
            Modelo(s1,s2).estado.T(1:end)=0;
        end
    end
end

estimaciones=0;
estimacion=1;
estados=[];
estimacion=[];
estimacion_final=[];

%Variables para el simulador
load('Angulos.mat');
[N_angulos,articulaciones]=size(Angulos);
cont=1; %Sera el que vaya graficando cada fila de angulos
empezando=0; %Controlará el transitorio de inicio
terminando=0; %Controlará el transitorio de fin
AngulosReposo=Angulos(N_angulos,:);
N_inicio=64; %Filas que corresponden al comienzo
N_fin=95; %Filas que corresponden al final
simulador(AngulosReposo,1,1);
velocidad=10; %En muestras por 8/128 seg.

OUTPUT=[];

fprintf('\n\nPreparado para obtener predicciones\n\n');
tic
while true

    %Nuevo estado cada 10 segundos
    if mod(estimaciones,20)==0
        estados(end+1)=randi([1,2]);
    end

    %Lectura cada 0.5 segundos
    if toc>=0.5

        %Lectura de datos de equipo
        state = calllib('edk','EE_EngineGetNextEvent',eEvent);
        eventType = calllib('edk','EE_EmoEngineEventGetType',eEvent);
        userID=libpointer('uint32Ptr',0);
        calllib('edk','EE_EmoEngineEventGetUserId',eEvent, userID);

        if strcmp(eventType,'EE_UserAdded') == true
            User_added = 1;
            userID_value = get(userID,'value');
            calllib('edk','EE_DataAcquisitionEnable',userID_value,true);
            readytocollect = true;
```

```
        end

        if (readytocollect)
            calllib('edk','EE_DataUpdateHandle', 0, hData);
            nSamples = libpointer('uint32Ptr',0);
            calllib('edk','EE_DataGetNumberOfSample',hData,nSamples);
            nSamplesTaken = get(nSamples,'value');
            samples_taken = samples_taken + nSamplesTaken;
            samplesSinceLastWindow = samplesSinceLastWindow +
nSamplesTaken;

            if (nSamplesTaken ~= 0)
                data = libpointer('doublePtr',zeros(1,nSamplesTaken));
                for i =
1:length(fieldnames(enuminfo.EE_DataChannels_enum))
                    calllib('edk','EE_DataGet',hData,
DataChannels.([DataChannelsNames{i}]), data, uint32(nSamplesTaken));
                    data_value = get(data,'value');
                    INPUT = data_value(i,[4:17]);
                end
            end
        end
        %Al final de este proceso, INPUT debería contener el buffer de
        %datos de 0.5 segundos de Emotiv EPOC

        %Laplaciana y PSD de la ventana
        [filas,columnas]=size(OUTPUT);
        [M,N]=size(INPUT'); %M=numero de sensores, N=numero de de datos
        OUTPUT=[OUTPUT LaplacianFilterZonesSetV(INPUT',x,y)];
        dato=[];
        for j= 1 : 1 : M
            dato1 = pwelch(OUTPUT(j,end-ventana+1:end) -
mean(OUTPUT(j,end-ventana+1:end)),[],[],F,Fs);
            dato = [dato dato1];
        end

        lista = 1 : 1 : length(dato);

        %Test y Predicciones
        %Test
        for s1=1 : 1 : 3
            for s2=1 : 1 : 3
                if s1 ~= s2
                    Modelo(s1,s2) =
predecir_SdFasArt_seleccion(Modelo(s1,s2),dato,lista,Modelo(s1,s2).parame
tros.delta,Modelo(s1,s2).parametros.At);
                    estimacion(end+1) = Modelo(s1,s2).salida.prediccion;
                end
            end
        end

        %Predicciones
        valor=mean(estimacion);
        if valor> 1.6 %Para que necesite como mínimo 4 de 6 marchas
            estimacion_final(end+1)=2;
        else
            estimacion_final(end+1)=1;
        end;
        if estimacion_final(end) ==  estado(end)
            aciertos = aciertos + 1;
```

```
        end
        etiqueta=estados(end);
        estimaciones = estimaciones + 1;
        estimacion=[];

        %Simulador de exoesqueleto
        %sincronizador para los pasos
        if cont >= N_angulos
            cont=cont-N_angulos+1;
        end
        if n>2 && simulador_on==1
            if estimacion_final(end)==2
                if (estimacion_final(end-1)==1 || empezando==1) &&
terminando==0;
                    if empezando==0
                        cont=1;
                    end
                    empezando=1;
                    %empezando
                    simulador(Angulos(cont,:),cont,estado);
                    Mo(cont)=getframe;
                    if cont>=N_inicio
                        empezando=0;
                    end
                end
                if estimacion_final(end-1)==2 && empezando==0 &&
terminando == 0;
                    %caminando
                    simulador(Angulos(cont,:),cont,estado);
                    Mo(cont)=getframe;
                    if cont >= N_angulos-N_fin
                        cont=N_inicio;
                    end
                end
            end
            if estimacion_final(end)==1
                if (estimacion_final(end-1)==2)
                    terminando=1;
                end
                if (cont<=N_angulos-N_fin-velocidad && cont >=velocidad)
&& terminando==1
                    %caminando
                    simulador(Angulos(cont,:),cont,estado);
                    Mo(cont)=getframe;
                end
                if (estimacion_final(end-1)==1 && terminando==0)
                    %parado
                    simulador(AngulosReposo,cont,estado);
                    Mo(cont)=getframe;
                end
                if (cont>N_angulos-N_fin-velocidad || cont <velocidad) &&
terminando==1
                    %terminando
                    simulador(Angulos(cont,:),cont,estado);
                    Mo(cont)=getframe;
                    if (cont>=N_angulos-velocidad || cont <=velocidad) &&
terminando==1
                        terminando=0;
                    end
                end
            end
```

```
        end
        cont=cont+velocidad;
    end
end

% close all;
plot(etiqueta);
hold on;
plot(estimacion_final,'*');

aciertos_marcha=0;
fallos_marcha=0;
aciertos_paro=0;
fallos_paro=0;
for i=1:length(etiqueta)
    if etiqueta_final(i)==1
        if estimacion_final(i)==1
            aciertos_paro=aciertos_paro+1;
        else
            fallos_paro=fallos_paro+1;
        end
    elseif etiqueta_final(i)==2
        if estimacion_final(i)==2
            aciertos_marcha=aciertos_marcha+1;
        else
            fallos_marcha=fallos_marcha+1;
        end
    end
end
porcentaje_marcha=aciertos_marcha/(aciertos_marcha+fallos_marcha)
porcentaje_paro=aciertos_paro/(aciertos_paro+fallos_paro)
```

Anexo V. Simulador de exoesqueleto

```
function [correcto] = simulador(angulos, f,estado)

lmuslo = 200; %Es la distancia que tendrá el muslo cuando se muestre por
pantalla
lpantorrilla = lmuslo ; %Es la distancia que tendrá la pantorrilla cuando
se muestre por pantalla
lcadera=lmuslo+lpantorrilla; %Es la distancia del los pies al suelo.
ltronco=lcadera*2; %Es la longitud del tronco
lpie = 70; %Es la longitud del pie, es decir el número que tiene

%---------------------Pierna Derecha-----------------
 X = zeros (1, 5);
 Y = zeros (1, 5);
 Z = zeros (1, 5);

  %% Posición Cabeza y tronco
    %cabeza y brazos
        X(1)=0;  %Indica la posición de la cabeza respecto al avance de
la pierna
        Y(1)=ltronco+3;  %Indica la posicion de la cabeza respecto a la
altura
        Z(1)=-100;  %Indica la posición de las lineas azules respecto a
la izquiera o derecha
    %tronco
        X(2)=0;  %Indica la posición del tronco respecto al avance
        Y(2)=lcadera;  %Indica la posicion del tronco respecto a la
altura
        Z(2)=-100;  %Indica la posición de las lineas azules respecto a
la izquiera o derecha
  %% Movimiento pierna
       % cadera y muslo
        Theta1 = angulos(3)-pi/2;
        X(3) = cos (Theta1)*lmuslo + X(2);
        Y(3) = sin (Theta1)*lmuslo + Y(2);
        Z(3)=-100;
    % Pantorilla y rodilla
        Theta2 =Theta1-angulos(2);
        X(4) = cos (Theta2)*lpantorrilla + X(3);
        Y(4) = sin (Theta2)*lpantorrilla + Y(3);
        Z(4)=-100;
    % Tobillo y pie
        Theta3 = Theta2-angulos(1)+pi/2;
        X(5) = cos (Theta3)*lpie+ X(4);
        Y(5) = sin (Theta3)*lpie + Y(4);
        Z(5)=-100;

%---------------------Pierna Izquierda-----------------
 X1 = zeros (1, 5);
 Y1 = zeros (1, 5);
 Z1 =zeros (1, 5);
 %%%%%
  %% Posición Cabeza y tronco
    %cabeza y brazos
        X1(1)=0;  %Indica la posición de la cabeza respecto al avance de
la pierna izquierda
        Y1(1)=ltronco+3;  %Indica la posicion de la cabeza respecto a la
altura
```

```
        Z1(1)=100;  %Indica la posición de las lineas azules respecto a
la izquiera o derecha
    %tronco
        X1(2)=0;  %Indica la posición del tronco respecto al avance
        Y1(2)=lcadera;  %Indica la posicion del tronco respecto a la
altura
        Z1(2)=100;  %Indica la posición de las lineas azules respecto a
la izquiera o derecha
   %% Movimiento pierna graficamente
    % cadera y muslo
        Theta4 = angulos(6)-pi/2;
        X1(3) = cos (Theta4) * lmuslo + X1(2);
        Y1(3) = sin (Theta4) * lmuslo + Y1(2);
        Z1(3)=100;
    % Pantorilla y rodilla
        Theta5 =Theta4-angulos(5);
        X1(4) = cos (Theta5) * lpantorrilla + X1(3);
        Y1(4) = sin (Theta5) * lpantorrilla + Y1(3);
        Z1(4)=100;
    % Tobillo y pie
        Theta6 = Theta5-angulos(4)+pi/2;
        X1(5) = cos (Theta6)*lpie+ X1(4);
        Y1(5) = sin (Theta6)*lpie+ Y1(4);
        Z1(5)=100;

 %% Graficación de los angulos
    %Pierna derecha
        Ra = 10;  %Hace que los angulos de la pierna derecha se muestren
a una determinada altura del plano verde
        plot3(X,Z,Y,'-or','MarkerFaceColor',[0 .75
.75],'LineWidth',2.0,'Color',[0 0.4470 0.7410]); %Es el color y grosor de
la pierna derecha
        hold on   %Hace que la pierna derecha siga mostrandose cuando se
compile otro Plo3 despues de este
        axis off  %No muestra las coordenadas de los ejes

        axis equal; %Hace que la representación de todo el muñeco se vea
equilibrada
        NumPoints = 20;
        Tz = zeros (1, NumPoints); %Rellena de ceros la linea 1 durante
NumPoints=20 veces

        if (angulos(3))
        DT1 = angulos(3)/(NumPoints-1);
        T1 = pi/2+[0:DT1:angulos(3)];
        else
        T1 = pi/2 + Tz;
        end
        T1x = X(2) + Ra * cos (T1); %Inidca el avance en el que se
muestra el ángulo de la cadera pierna derecha
        T1y = Y(2) + Ra * sin (T1); %Indica la altura a la que se muestra
el ángulo de la cadera pierna derecha

        if (angulos(2))
        DT2 = angulos(2)/(NumPoints-1);
        T2 = pi/2 + angulos(1) + angulos(2) -[0:DT2:angulos(2)];
        else
        T2 = pi/2 + angulos(1) + angulos(2) + Tz;
        end
```

```
        T2x = X(3) + Ra * cos (T2); %Inidca el avance en el que se
muestra el ángulo de la rodilla pierna derecha
        T2y = Y(3) + Ra * sin (T2); %Indica la altura a la que se muestra
el ángulo de la rodilla pierna derecha

        if (angulos(1))
        DT3 = angulos(1)/(NumPoints-1);
        T3 = pi/2 + angulos(1) + angulos(2) + angulos(3) -
[0:DT3:angulos(1)];
        else
        T3 =angulos(1) + angulos(2) + angulos(3)+ Tz;
        end
        T3x = X(4) + Ra * cos (T3); %Inidca el avance en el que se
muestra el ángulo del tobillo pierna derecha
        T3y = Y(4) + Ra * sin (T3); %Indica la altura a la que se muestra
el ángulo del tobillo pierna derecha

    %Pierna izquierda
        Ra = 10;  %Hace que los angulos de la pierna izquierda se
muestren a una determinada altura del plano verde
        plot3(X1,Z1,Y1,'-or','MarkerFaceColor',[1 1
0],'LineWidth',1.5,'Color',[0.9 0.8470 0]); %Es el color y grosor de la
pierna izquierda
        NumPoints = 20;
        Tz = zeros (1, NumPoints);

        if (angulos(6))
        DT4 = angulos(6)/(NumPoints-1);
        T4 = pi/2+[0:DT4:angulos(6)];
        else
        T4 = pi/2 + Tz;
        end
        T4x = X1(2) + Ra *100* cos (T4); %Inidca el avance en el que se
muestra el ángulo de la rodilla pierna izquierda
        T4y = Y1(2) + Ra * sin (T4); %Indica la altura a la que se
muestra el ángulo de la cadera pierna izquierda

        if (angulos(5))
        DT5 = angulos(5)/(NumPoints-1);
        T5 = pi/2 + angulos(4) + angulos(5) -[0:DT5:angulos(5)];
        else
        T5 = pi/2 + angulos(4) + angulos(5) + Tz;
        end
        T5x = X1(3) + Ra * cos (T5); %Inidca el avance en el que se
muestra el ángulo de la rodilla pierna izquierda
        T5y = Y1(3) + Ra * sin (T5); %Indica la altura a la que se
muestra el ángulo de la rodilla pierna izquierda

        if (angulos(4))
        DT6 = angulos(4)/(NumPoints-1);
        T6 = pi/2 + angulos(4) + angulos(5) + angulos(6) -
[0:DT6:angulos(4)];
        else
        T6 =angulos(4) + angulos(5) + angulos(6)+ Tz;
        end
        T6x = X1(4) + Ra * cos (T6); %Inidca el avance en el que se
muestra el ángulo del tobillo pierna izquierda
        T6y = Y1(4) + Ra * sin (T6); %Indica la altura a la que se
muestra el ángulo del tobillo pierna izquierda
 %% Graficación de los elementos
```

```
     %suelo
        x = [-400 -400 400 400];  %Largo del terreno
        y = [-250 250 250 -250];  %Ancho del terreno
        z = [-15 -15 -15 -15];    %Altura del terreno
        if estado==1
            patch(x,y,z,'White')
        elseif estado==2
            patch(x,y,z,'green')
        end

     %Cabeza
        BoxPlot3(X(1)-38, -40, Y(1)+22, 80,80,80) %Posición y tamaño de
la cabeza

     %Tronco
        BoxPlot3(X(2)-18, -115, Y(2)+12, 35,230, ltronco-lcadera+10)
%Posición y tamaño del tronco

     %Mano Derecha
        BoxPlot3(X(1), -130, Y(1), 15,15, -(ltronco-lcadera-20))
%Posición y tamaño de la mano derecha

     %Mano Izquierda
        BoxPlot3(X1(1), 130, Y1(1), 15,15, -(ltronco-lcadera-20)/2)
%Posición y tamaño de la parte superior de la mano izquierda
        BoxPlot3(X1(1), 130, Y1(1)-(ltronco-lcadera-20)/2,(ltronco-
lcadera-20)/2 ,15,15 ) %Posición y tamaño de la parte inferior de la mano
izquierda

 %% Texto y formato de los angulos
        axes1 = gca; %Muestra los angulos
  %Pierna derecha
      %Cadera derecha
        strText = sprintf ('$\\theta_{1} (%2.2f grad)$',angulos(1)*180 /
pi);
        text('FontName', 'Arial', 'FontSize', 9,'FontWeight','bold',
'Position',[T1x(NumPoints/2)+150,Tz(NumPoints/2)-
235,T1y(NumPoints/2)+55],'String',strText,'interpreter','latex','Horizont
alAlignment','center','Parent',axes1);
      %Rodilla derecha
        strText = sprintf ('$\\theta_{2} (%2.2f grad)$', angulos(2) * 180
/ pi);
        text('FontName', 'Arial', 'FontSize', 9,'FontWeight','bold',
'Position', [T2x(NumPoints/2)+150,Tz(NumPoints/2)-
235,T2y(NumPoints/2)+55], 'String', strText,'interpreter', 'latex',
'HorizontalAlignment', 'center','Parent', axes1);
      %Tobillo derecha
        strText = sprintf ('$\\theta_{3} (%2.2f grad)$', angulos(1) * 180
/ pi);
        text('FontName', 'Arial', 'FontSize', 9,'FontWeight','bold',
'Position', [T3x(NumPoints/2)+150,Tz(NumPoints/2)-
235,T3y(NumPoints/2)+55], 'String', strText,'interpreter', 'latex',
'HorizontalAlignment', 'center','Parent', axes1);

  %Pierna izquierda
      %Cadera izquierda
        strText = sprintf ('$\\theta_{4} (%2.2f grad)$', angulos(6) * 180
/ pi);
        text('FontName', 'Arial', 'FontSize', 9,'FontWeight','bold',
'Position', [T4x(NumPoints/2)-
```

```
170,Tz(NumPoints/2)+200,T4y(NumPoints/2)]+15, 'String',
strText,'interpreter', 'latex', 'HorizontalAlignment', 'center','Parent',
axes1);
      %Rodilla izquierda
        strText = sprintf ('$\\theta_{5} (%2.2f grad)$', angulos(5) * 180
/ pi);
        text('FontName', 'Arial', 'FontSize', 9,'FontWeight','bold',
'Position', [T5x(NumPoints/2)-
170,Tz(NumPoints/2)+200,T5y(NumPoints/2)]+15, 'String',
strText,'interpreter', 'latex', 'HorizontalAlignment', 'center','Parent',
axes1);
      %Tobillo izquierdo
        strText = sprintf ('$\\theta_{6} (%2.2f grad)$', (angulos(4) *
180 / pi));
        text('FontName', 'Arial', 'FontSize', 9,'FontWeight','bold',
'Position', [T6x(NumPoints/2)-
170,Tz(NumPoints/2)+200,T6y(NumPoints/2)]+15, 'String',
strText,'interpreter', 'latex', 'HorizontalAlignment', 'center','Parent',
axes1);

hold off %Hace que cada angulo vaya eliminandose una vez se genera el
siguiente
return

end
```