UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESARROLLO DE UNA INTERFAZ HOMBRE-MÁQUINA BASADA EN SEÑALES ELECTROOCULOGRÁFICAS (EOG)

GRADO EN INGENIERÍA DE
TECNOLOGÍAS DE TELECOMUNICACIÓN

ALBA SIERRA DELGADO
MÁLAGA, 2024

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

Desarrollo de una Interfaz Hombre-Máquina Basada en Señales Electrooculográficas (EOG)

Autor: Alba Sierra Delgado

Tutor: Ricardo Ron Angevin

Departamento: Tecnología Electrónica

Titulación: Grado en Ingeniería de Tecnologías de Telecomunicación

Palabras clave: Señal Electrooculográfica, Movimiento Ocular Sacádico, Interfaz Hombre Máquina, Tecnología Adaptativa, Máquina de Estados, MATLAB

Resumen

Este trabajo aborda el desarrollo de una interfaz hombre-máquina basada en señales EOG (*Electrooculográficas*), destinada a controlar dispositivos mediante los movimientos oculares. El objetivo principal fue diseñar una interfaz que interpretase las señales para manipular un cursor en pantalla, facilitando así el acceso a la tecnología para personas con discapacidades motoras. Utilizando MATLAB, se implementaron algoritmos específicos para el procesamiento de las señales recibidas, logrando traducirlas en comandos en tiempo real. Los resultados demostraron que las señales EOG pueden controlar eficazmente un cursor, confirmando la precisión y capacidad de respuesta del sistema. Este proyecto no solo valida la viabilidad de las señales EOG para el control de dispositivos, sino que también sienta las bases para desarrollos futuros en interfaces hombre-máquina, con el potencial de mejorar significativamente la calidad de vida de las personas con limitaciones físicas.

E.T.S. DE INGENIERÍA DE TELECOMUNICACIÓN, UNIVERSIDAD DE MÁLAGA

Development of a Human-Machine Interface Based on Electrooculographic (EOG) Signals

Author: Alba Sierra Delgado

Supervisor: Ricardo Ron Angevin

Department: Electronic Technology
Degree: Bachelor of Telecommunication Technology Engineering

Keywords: Electrooculographic Signal, Saccadic Eye Movement, Human-Machine Interface, Adaptive Technology, State Machine, MATLAB

Abstract

The purpose of this project is the development of a human-machine interface based on EOG (*Electrooculographic*) signals, aimed at controlling devices through eye movements. The main objective was to design an interface that interprets the signals to manipulate an on-screen cursor, thus facilitating access to technology for people with motor disabilities. Using MATLAB, specific algorithms were implemented to process the received signals and translate them into real-time commands. The results demonstrated that EOG signals can effectively control a cursor, confirming the accuracy and responsiveness of the system. This project not only validates the feasibility of EOG signals for device control, but also lays the groundwork for future developments in human-machine interfaces, with the potential to significantly improve the quality of life of people with physical limitations.

Agradecimientos

A los profesores, como mi tutor, Ricardo, que se apasionan por su trabajo y consiguen transmitir esa pasión a sus alumnos.

A mis padres y mi familia, por siempre empujarme a seguir estudiando y despertar esta curiosidad tan grande que tengo por aprender.

A mis compañeros y amigos de la universidad, por los apuntes, las dudas y las tardes de biblio. No habría pasado del primer curso sin vosotros.

A Nacho, a Laura y a todos los que me aguantáis hablar de teleco como si os interesase, por estar ahí y quedaros a apoyarme incondicionalmente.

A mi familia.

Índice

Acrónimos

DAQ	Data Acquisition
EOG	Electrooculográfica
ETSIT	Escuela Técnica Superior de Ingeniería de Telecomunicación
HMI	Human Machine Interface
MO	Movimiento Ocular
TA	Tecnología Asistiva
TFG	Trabajo de Fin de Grado
TIC	Tecnologías de la Información y las Comunicaciones
UMA	Universidad de Málaga

Índice de figuras, códigos y tablas

Capítulo 1. Introducción

1.1. Marco de referencia

El papel de las TIC (*Tecnologías de la Información y la Comunicación*) en nuestra sociedad contemporánea es indiscutible. Desde la comunicación hasta el trabajo, pasando por el entretenimiento y la educación, las TIC se han integrado profundamente en nuestra vida diaria, transformando la forma en que interactuamos, aprendemos y nos conectamos con el mundo que nos rodea. En este sentido, la inclusión y la accesibilidad son principios fundamentales que deben tenerse en cuenta a la hora de implementar estas tecnologías, asegurando que todas las personas, independientemente de sus capacidades o limitaciones, puedan beneficiarse de ellas.

En particular, para las personas con discapacidad, las TIC pueden representar tanto una oportunidad como un desafío. Si bien estas tecnologías tienen el potencial de mejorar significativamente la calidad de vida y la participación en la sociedad, también pueden presentar barreras significativas de accesibilidad que dificultan su uso. Es aquí donde entran en juego las TA (*Tecnologías Asistivas*), que desempeñan un papel crucial en la eliminación de estas barreras y la inclusión digital para todos.

En el contexto de las personas con discapacidad motora, como la parálisis, el seguimiento ocular ha surgido como una herramienta vital para mejorar la accesibilidad tecnológica. En [1], se destaca cómo los periféricos estándar de los ordenadores y otros dispositivos de acceso a las TIC pueden resultar inaccesibles para las personas con parálisis cerebral, debido a las exigencias de movilidad precisa, coordinación, nivel cognitivo y agudeza visual que estos elementos requieren. En respuesta a esta necesidad, el desarrollo de una HMI (*Interfaz Hombre-Máquina, del inglés Human-Machine Interface*) que facilite el control de dispositivos mediante el seguimiento ocular se presenta como una solución prometedora. Al permitir que las personas con discapacidad motora controlen dispositivos tecnológicos utilizando simplemente el movimiento de sus ojos, esta HMI no solo mejora la accesibilidad, sino que también promueve la independencia y la autonomía.

En [2] se destacan tres tipos fundamentales de dispositivos para el registro del MO (*Movimiento Ocular*), dependiendo de su nivel de invasividad.

El primero, basado en un primer diseño de Edmund Huey, implica la colocación de una lente de contacto con un agujero, en el ojo del sujeto. Aunque ofrece una precisión detallada en el registro del MO, resulta un método muy invasivo y la molestia que puede generar en los participantes hace que no sea la opción más ideal.

Posteriormente, Guy Thomas Buswell ideó el primer dispositivo de seguimiento ocular no invasivo utilizando haces de luz reflejados en el ojo. Hoy en día existen multitud de dispositivos, como por ejemplo el de la figura 1.1, que utilizan luz infrarroja, cámaras de vídeo o sensores ópticos para detectar el movimiento de la pupila. Estos dispositivos, más en concreto los que usan luz infrarroja, son los comúnmente llamados "*eye-tracker"*. Aunque su costo es bajo y no requiere contacto directo con el ojo, pueden presentar dificultades para detectar con precisión la posición ocular. Además, no es posible registrar la conducta visual cuando los ojos están cerrados.

Figura 1.1. Hiru hardware, IRISBOND eye-tracking. Fuente: [3]

Finalmente, el método basado en potenciales eléctricos ofrece una alternativa intrigante. Utilizando electrodos colocados alrededor de los ojos, este método registra el campo de potencial eléctrico generado por el MO, conocido como electrooculograma (EOG). Una de sus principales ventajas radica en su capacidad para registrar el MO incluso con los ojos cerrados, ya que lo que detecta es el campo eléctrico del ojo. Aunque implica cierto grado de invasividad al requerir la colocación de electrodos, como puede observarse en la figura 1.2., su capacidad para funcionar incluso con los párpados cerrados y su relativa independencia de la precisión de la detección de la pupila y la córnea lo convierten en una opción atractiva para el uso en TA.

Figura 1.2. Persona utilizando un lector de EOG. Fuente: [4]

Es por esto, por lo que el desarrollo de una HMI que permita utilizar los datos obtenidos mediante un dispositivo de lectura de EOG para controlar el movimiento a derecha e izquierda o arriba y abajo de un dispositivo como podría ser un cursor, supone una buena alternativa.

1.2. Objetivo

En el grupo de investigación DIANA perteneciente al departamento de Tecnología Electrónica de la ETSIT (Escuela *Técnica Superior de Ingeniería de Telecomunicación*) de la UMA (*Universidad de Málaga*) se dispone de un dispositivo capaz de registrar señales EOG que, unido a una tarjeta DAQ (*Adquisición de Datos, del inglés Data Acquisition*), permite su procesado mediante MATLAB.

El objetivo de este TFG (*Trabajo de Fin de Grado*) es el desarrollo de una interfaz hombre-máquina compatible con este dispositivo. La interfaz debe permitir, mediante el uso de señales EOG, controlar un dispositivo.

Para conseguirlo y constatar que el programa funciona correctamente, se han desarrollado una serie de funciones en MATLAB. Las funciones permiten, mediante el procesado de los datos recibidos a través de la tarjeta DAQ, interpretar ciertos MO del usuario como comandos. Estos comandos se ven traducidos en el movimiento de arriba, abajo, izquierda o derecha de un cursor mostrado en pantalla.

1.3. Estado del arte

El uso de dispositivos de seguimiento ocular está a la orden del día. Cada vez son más las empresas que ofrecen dispositivos capaces de interpretar el MO y utilizarlo como controlador. Es el caso por ejemplo de las Apple Vision Pro [5], que se presentan como una nueva forma de integrar la tecnología en nuestro día a día. Sin embargo, la mayoría de estos dispositivos que salen al mercado realizan el seguimiento ocular mediante cámaras de luz infrarroja o LED (*Diodo Emisor de Luz, del Inglés Light Emmiting Diode*) mientras que el uso de señales EOG solo se ha extendido de momento al ámbito académico y de investigación.

En [6], publicado en 2023, se describe el desarrollo de una HMI que utiliza un lector de señales EOG. Este dispositivo, tras el procesamiento de los datos, transmite las instrucciones a un dispositivo controlado por Arduino a través de señales Bluetooth. Estas instrucciones se utilizan para controlar el movimiento de una silla de ruedas.

En [7], publicado en marzo de 2024, el trabajo se enfoca en asociar cada MO obtenido mediante señales EOG a una frase concreta. Este enfoque resultó en la creación de una interfaz diseñada para facilitar la comunicación del usuario.

Previamente, en mayo de 2018, se publicó [8] un estudio relevante en el cual, en contraposición a los enfoques previamente mencionados que consideraban el MO, se analizaron las señales derivadas del parpadeo del usuario. Según este estudio, se logró de manera efectiva que los usuarios pudieran controlar el entorno de una vivienda inteligente mediante la interacción con una interfaz que requería el registro de sus parpadeos como señales EOG.

Es evidente que la creación de una HMI que utilice las señales EOG para controlar un dispositivo ofrece diversas y útiles posibilidades para mejorar la calidad de vida de las personas, ya que, mediante un procesamiento adecuado, es posible controlar casi cualquier dispositivo

1.4. Estructura de la memoria

Este TFG está organizado en cinco capítulos. El primero constituye una introducción destinada a situar al lector en el contexto del ámbito de aplicación del proyecto. Funciona como

un marco de referencia que resalta la importancia del desarrollo de interfaces controladas mediante señales EOG, así como la situación actual de investigaciones similares. Además, en este capítulo se detallan los objetivos y el propósito del proyecto.

El segundo capítulo tiene como objetivo proporcionar toda la información necesaria para comprender el TFG. Se explica qué son y cómo funcionan las señales EOG así como sus aplicaciones en HMI. Además se proporciona información sobre los recursos de los que se dispone para la captura y el análisis de estas señales.

El tercer capítulo, el principal, se centra en la definición e implementación de los requisitos y funcionalidades de la interfaz. Aquí se explica detalladamente el proceso completo, desde la interpretación de los datos recopilados hasta el desarrollo del código en MATLAB para cumplir con las expectativas del proyecto. Se resaltan las pruebas y ajustes realizados, así como la integración con el dispositivo de salida y el usuario final.

En el cuarto capítulo se describen los métodos utilizados y las pruebas realizadas para evaluar el correcto funcionamiento de la interfaz. Se incluyen los resultados obtenidos de estos experimentos, así como su interpretación.

Finalmente, se agrega un capítulo de conclusiones que resume los resultados obtenidos y su alcance. Además, se ofrecen ideas sobre futuras aplicaciones y posibles mejoras de la interfaz.

Capítulo 2. Marco teórico

2.1. Introducción

En este segundo capítulo se estudian los fundamentos teóricos necesarios para entender qué son y para qué sirven las señales EOG y cómo estas pueden ser utilizadas para el control de dispositivos. Para comenzar, se indaga sobre los fundamentos y las características de las señales EOG, permitiendo un entendimiento más profundo de ellas, lo que ayudará a la hora de definir la HMI. En la segunda sección, se realiza una breve descripción sobre las posibles aplicaciones de integrar estas señales en una HMI. Finalmente, se mencionan los recursos de los que dispone el laboratorio de la ETSIT de la UMA para la captura y análisis de estas señales.

2.2. Fundamentos y características de las señales EOG

El estudio y la utilización de las señales electrooculográficas se fundamenta en la idea de considerar el ojo como un dipolo, lo que permite identificar sus movimientos a partir de las diferencias de potencial registradas. Según los estudios, dentro del globo ocular existe una diferencia de potencial notable entre la córnea y la retina, regiones del ojo señaladas en la Figura 2.1. Esta diferencia de potencial posibilita que el ojo pueda ser considerado como un dipolo. Como se detalla en [9], el campo eléctrico generado por este dipolo puede registrarse colocando una serie de electrodos alrededor de la órbita. El valor y la polaridad del potencial generado por el dipolo en un momento dado dependen del ángulo formado con respecto a los electrodos, lo que facilita una relación precisa entre las variaciones del potencial registrado y los MO.

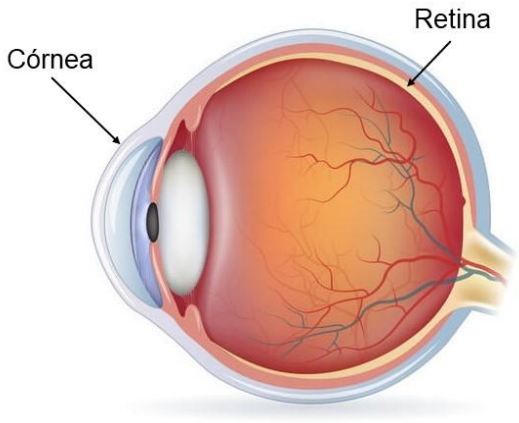

Figura 2.1 Situación de la córnea y la retina en el ojo. Fuente: [10]

Los movimientos sacádicos, que consisten en desplazamientos rápidos de los ojos entre dos puntos de fijación, son de particular interés para la aplicación que se persigue en este TFG.

Estos movimientos consisten en "*desplazamientos rápidos de los ojos entre dos puntos de fijación*" [9], es decir, son aquellos movimientos que permiten fijar la vista en diferentes puntos del espacio y enfocar a lo que se está mirando. Lo interesante de los movimientos sacádicos es que son movimientos que el ser humano realiza de forma voluntaria. Al usarse para cambiar la mirada desde un punto de interés actual a otro fuera del campo visual del usuario, la variación de potencial registrada es significativa y de corta duración debido a la rapidez con la que se realizan. En el esquema representado en la figura 2.2. se especifica cómo se interpretaría un movimiento sacádico horizontal si los electrodos se encontrasen situados en el canto interno y externo del ojo.

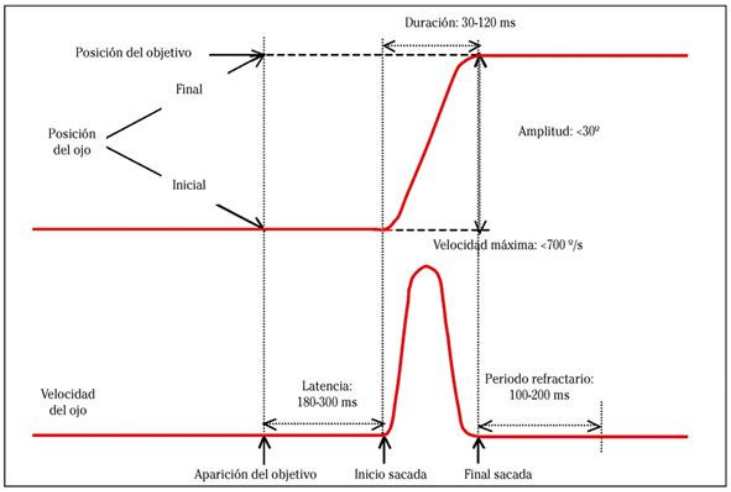

Figura 2.2 Esquema de un registro oculográfico de un movimiento sacádico horizontal. Fuente: [9]

Si se define como potencial positivo el acto de mirar a la izquiera, esto se vería representado en la señal como un valor de potencial positivo. El amplificador del que se dispone en este trabajo elimina la componente continua de la señal. Esto implica que, aunque el usuario mantenga la mirada fija en un punto a la izquierda, la señal que se obtendrá tendrá la forma de un pico que regresa a cero en lugar de una constante positiva. Esto implica que el acto de fijar la mirada en un punto a la derecha y después volver a dirigirla al centro, se registre como un pico de potencial negativo seguido de otro pico de potencial positivo.

Es importante recalcar que, un movimiento de fijación de la mirada hacia un punto más alejado respecto a uno más cercano, se traduce en una mayor diferencia de potencial entre el valor máximo de la señal y el valor de referencia. Esto se puede observar en las Figuras 2.3 y 2.4. Ambas figuras representan una serie de movimientos a la derecha seguidos de fijar la vista de nuevo en el centro de la pantalla, sin embargo, los movimientos de la figura 2.3 se mantuvieron dentro de los límites de la pantalla del ordenador, mientras que los de la figura 2.4 se realizaron de forma más exagerada fijando la mirada en un punto más alejado a la derecha.

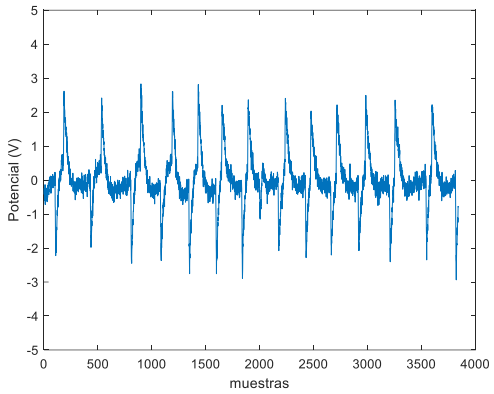

Figura 2.3 Señal EOG de movimientos cortos a la derecha y vuelta al frente (MATLAB)

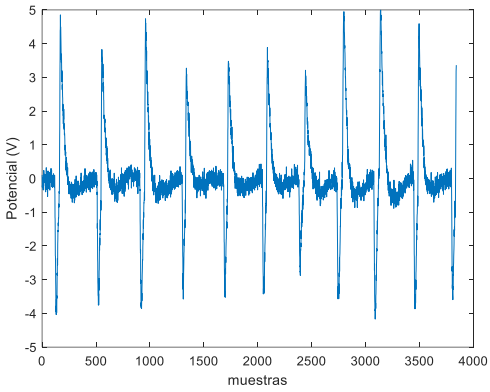

Figura 2.4 Señal EOG de movimientos amplios a la derecha y vuelta al frente (MATLAB)

2.3. Aplicaciones de las señales EOG en interfaces hombre-máquina

El uso de señales EOG ha estado tradicionalmente reservado a la medicina, la neurociencia y la psicología. Sin embargo, existe una amplia variedad de aplicaciones para las EOG en el desarrollo de HMI que aprovechan su capacidad para registrar los movimientos oculares y convertirlos en comandos que controlan dispositivos. Algunas de las aplicaciones más destacadas incluyen:

- **Control de dispositivos electrónicos:** Las señales EOG pueden ser utilizadas para controlar dispositivos electrónicos como ordenadores, teléfonos móviles, televisores y sistemas de domótica. Los MO pueden traducirse en comandos que realizan acciones específicas, como desplazarse por menús, seleccionar elementos en la pantalla, escribir texto [7] o controlar los dispositivos de un hogar inteligente [8].
- **Asistencia para personas con discapacidad:** Las señales EOG ofrecen una forma alternativa de interacción para personas con discapacidades motoras graves, como la parálisis o la tetraplejia. Estas personas pueden usar movimientos oculares para comunicarse, acceder a dispositivos electrónicos o controlar dispositivos de asistencia como una silla de ruedas [6].
- **Interfaces de realidad virtual aumentada:** Las señales EOG se integran en interfaces de realidad virtual y aumentada para ofrecer una experiencia de usuario más inmersiva y natural [11]. Los MO pueden controlar la dirección de la mirada dentro del entorno virtual, permitiendo interacciones más intuitivas y realistas.

En resumen, las aplicaciones de señales EOG en HMI abarcan un amplio abanico de posibilidades, ofreciendo soluciones innovadoras para mejorar la interacción entre humanos y tecnología.

2.4. Recursos disponibles para la captura y análisis de señales EOG

El estudio y análisis de las señales EOG requiere el uso de dispositivos especializados y herramientas de análisis de datos. En esta sección se presentan los recursos disponibles en el laboratorio de la ETSIT para la captura y análisis de señales EOG que se utilizaron en este TFG.

2.4.1. Dispositivos de captura de señales EOG

Existen varios dispositivos en el mercado diseñados específicamente para la captura de señales EOG. Los utilizados en este proyecto son los siguientes:

- **Bsamp Biosignal Amplifier.** Este amplificador equipado con 16 canales permite la grabación simultánea de señales EEG, EMG, EOG y ECG. Sus módulos de canales seleccionables por el usuario facilitan la grabación tanto referenciada como verdaderamente bipolar, lo que permite una amplia gama de aplicaciones en campos que van desde la investigación en laboratorios hasta la rehabilitación [12]. Es importante recalcar, para una mejor compresión del proyecto, que el amplificador elimina la componente continua de las señales que graba. El aspecto de este aparato se puede apreciar en la figura 2.5.

Figura 2.5 g.Bsamp Biosignal Amplifier. Fuente: [12]

- **USB-6210 de National Instruments.** Este dispositivo de adquisición de datos USB (*Universal Serial Bus*) ofrece una alta velocidad de muestreo y múltiples canales de entrada analógica, lo que lo hace adecuado para la captura de señales EOG. Con su interfaz fácil de usar y su compatibilidad con software de análisis de datos, el USB-6210 es una opción versátil para el uso en HCI[13]. El aspecto de este aparato se puede apreciar en la figura 2.6.

Figura 2.6 USB-6210 de National Instruments. Fuente: [13]

2.4.2. Software de análisis de señales EOG

Para el análisis de las EOG capturadas en este TFG se utilizó, principalmente, MATLAB R2013a. El hecho de utilizar esta versión de MATLAB se debe a que, en este proyecto, se parte de unos códigos proporcionados por el grupo de investigación DIANA que contienen funciones incompatibles con versiones posteriores. MATLAB ofrece herramientas y funciones específicas para el procesamiento y análisis de señales biomédicas, incluidas las señales EOG. Con su amplia gama de funciones de procesamiento de señales, es una opción popular entre los investigadores para el análisis de datos EOG.

El software MATLAB, junto con los dispositivos de captura mencionados anteriormente y algunos cables y electrodos, comprenden la totalidad de elementos utilizados en este TFG con el objetivo de implementar la HCI que permita desplazamientos a derecha, izquierda, arriba y debajo de un cursor.

Capítulo 3. Diseño y desarrollo de la interfaz hombre-máquina

3.1. Introducción

En este capítulo se habla sobre el diseño y desarrollo de la HMI. Para tener una mejor idea del proceso de creación de esta interfaz, es importante conocer sus requisitos y funcionalidades. En la primera sección, se define detalladamente qué aspectos son importantes para el correcto funcionamiento de la HMI, proporcionando así una base sólida para su desarrollo y evaluación. La segunda sección trata sobre la implementación de la interfaz utilizando MATLAB, una herramienta muy utilizada en la ingeniería y la investigación. Una vez plasmado el código, se llevan a cabo los ajustes necesarios para garantizar su correcto funcionamiento. En esta etapa se evalúan la precisión y fiabilidad de la interfaz, y se corrigen posibles errores o fallos en su funcionamiento. Finalmente, se habla sobre la integración de la interfaz en tiempo real y la retroalimentación visual para el usuario.

3.2. Definición de la interfaz y sus requisitos de uso

Para la realización de este proyecto se parte de dos funciones MATLAB otorgadas por el grupo de investigación DIANA, *Medir_mov_completo* y *prueba_mov_completo*.

El programa *Medir_mov_completo* se encarga de la configuración de la tarjeta de adquisición de datos. Para ello se deben definir varios parámetros. Uno de los parámetros más importantes a configurar es la frecuencia de muestreo, que en este caso se ha establecido en 128 muestras por segundo.

En este programa se define también la variable *tam* o tamaño de bloque. Cada vez que llega un número de muestras igual al valor de *tam* se llama al programa *prueba_mov_completo*, que también recibe este valor como parámetro y se encarga del almacenaje y procesado de las muestras. Cuanto más pequeño sea el valor de *tam*, más rápido se realizará el procesado de las muestras. Un procesado rápido de las muestras es fundamental en aplicaciones que permiten operar en tiempo real, ya que permite una mejor experiencia de usuario y una mayor compatibilidad con otros dispositivos. En este proyecto se estableció el valor del tamaño de bloque en 16 muestras. Dado que la frecuencia de muestreo es de 128 muestras por segundo, esto quiere decir que la llamada a la función *prueba_mov_completo* y, por tanto, el procesado de las muestras se realiza cada 125 milisegundos.

El objetivo de este TFG es el de crear una función en MATLAB que pueda ser llamada por la función *prueba_mov_completo*. Esta función se encargará de procesar las señales obtenidas e identificar cuándo se produce un movimiento y en qué dirección. Para que esta función funcione correctamente, hay ciertos aspectos a considerar. En primer lugar, se debe definir qué MO se clasificarán como desplazamientos en las direcciones deseadas, izquierda, derecha, arriba o abajo. Esta definición asegura que la interfaz pueda reaccionar de manera adecuada a las intenciones del usuario. Por otro lado, no solo es importante la naturaleza de estos movimientos sino también la velocidad o amplitud con la que deben realizarse. Esto ayuda a que sean reconocidos por la interfaz y puedan distinguirse del ruido ocasionado por pequeños movimientos involuntarios del ojo o interferencias externas. A continuación se definen con detalle estos requisitos.

3.2.1. Naturaleza del movimiento ocular

Como se explica en la sección 2.2 de este documento, los movimientos sacádicos de los ojos producen una señal EOG muy concreta. Si el usuario define como posición inicial un punto en el centro de la pantalla y, a continuación, realiza un MO sacádico hacia la derecha, la señal EOG consistirá en un pico de voltaje negativo. Cuando el usuario vuelva a centrar su mirada en el punto inicial, se producirá un pico de voltaje positivo. La representación de un movimiento así en MATLAB es la de la figura 3.1. El hecho de observar el pico negativo cuando el MO es hacia la derecha depende de la posición en la que se coloquen los electrodos alrededor de los ojos. Para este proyecto se han colocado siempre los electrodos negativos a la derecha y en la parte superior del ojo y los positivos a la izquierda y en la parte inferior del ojo.

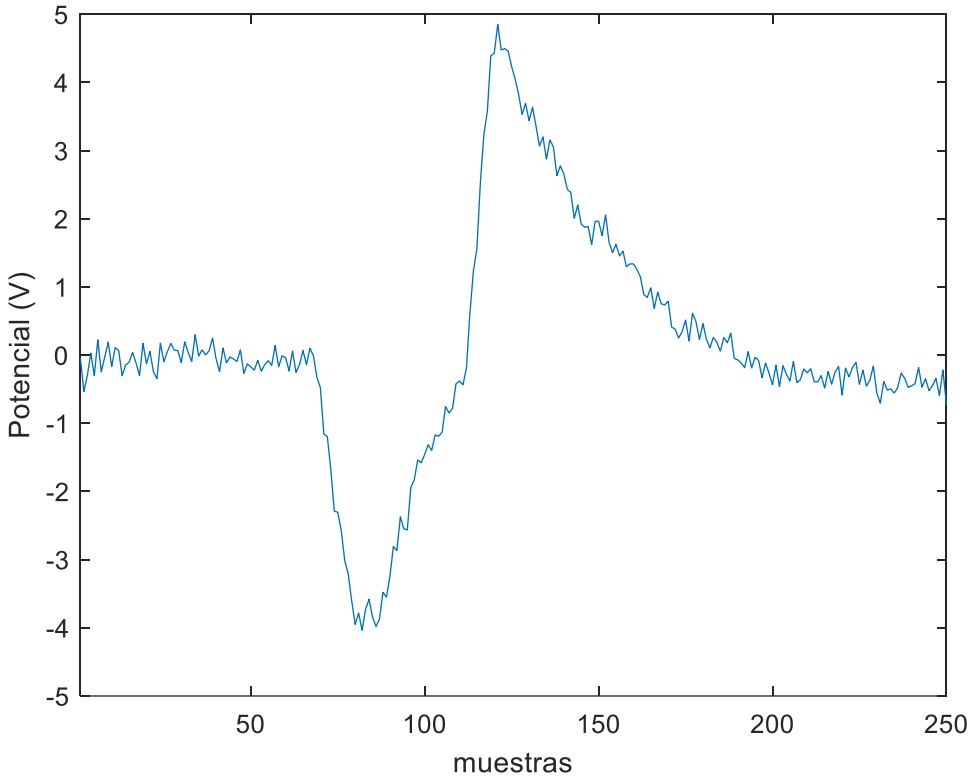

Figura 3.1 Movimiento a la derecha seguido de movimiento al centro (EOG)

Este movimiento que se ha definido será el que se interprete como el comando de mover a la derecha. Es un movimiento lo suficientemente concreto como para ser diferenciado del resto

de movimientos del ojo. Un movimiento inverso a este, primero con pico positivo y luego con pico negativo, será el comando de mover a la izquierda. Para los movimientos arriba y abajo se usará otro canal y la definición de arriba será la misma que la de derecha y la de abajo la misma que la de izquierda.

3.2.2. Velocidad del movimiento ocular

Para que el comando sea correctamente identificado y no se confunda o se mezcle con otro comando realizado a continuación, se deben definir una serie de requisitos en cuanto a la velocidad y duración del movimiento. En las figuras 3.2 y 3.3 se presentan dos posibles comandos de movimiento a la derecha. En el primero, el usuario mantiene la mirada en el punto de desplazamiento por un instante antes de regresar al punto inicial. Mientras que en la segunda, el movimiento de regreso es casi inmediato tras el movimiento a la derecha.

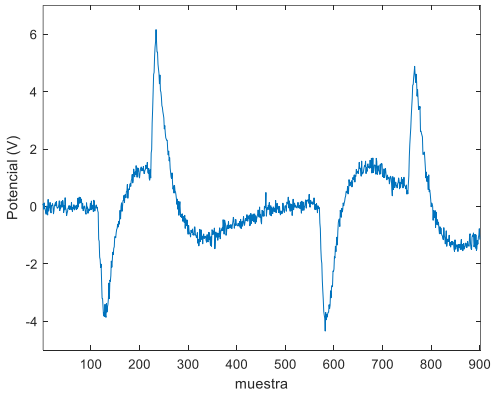

Figura 3.2 Comandos de movimiento a la derecha manteniendo la mirada en el punto alejado

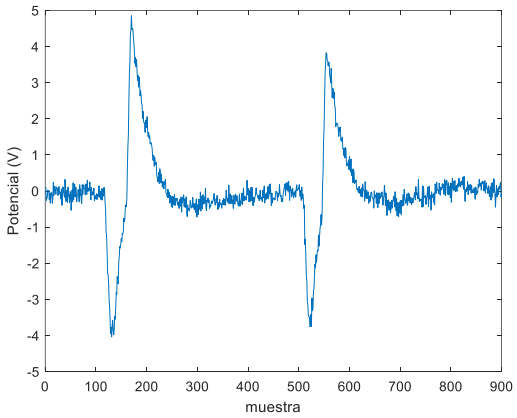

Figura 3.3 Comandos de movimiento a la derecha sin mantener la mirada en el punto alejado

Cuando la interfaz deba procesar estas señales, se debe definir un número concreto de muestras mínimo que debe haber entre un comando y el siguiente, además de un número de muestras máximo que debe haber entre el MO hacia la dirección deseada y el de vuelta. Si la señal a procesar es la de la figura 3.2, el número de muestras que hay entre el pico negativo y el positivo que pertenecen al mismo comando es muy cercano al número de muestras que hay entre el fin del primer comando (muestra 370 aproximadamente) y el inicio del segundo (muestra 570 aproximadamente). Esto podría ocasionar que la interfaz interpretase las muestras desde el instante 200 hasta el instante 700 como un movimiento hacia la izquierda, ya que se produce un pico positivo seguido de un pico negativo. Sin embargo, en la figura 3.3, los umbrales se encuentran mucho más diferenciados, lo que permitiría establecer un tiempo máximo entre movimientos más ajustado.

En conclusión, para que un comando sea correctamente identificado por la interfaz, los movimientos de mirar en la dirección deseada y de vuelta deben ser rápidos y simultáneos (0,5 segundos aproximadamente). Mientras que, el tiempo que se debe esperar entre un comando y otro debe ser suficiente para que la interfaz interprete que el comando anterior ha finalizado antes de analizar el siguiente (dos segundos aproximadamente).

3.2.3. Amplitud del movimiento ocular

La amplitud de la señal EOG es directamente proporcional a la amplitud en el MO. Es decir, cuanto más hacia la derecha mire el usuario, más alto será el pico de voltaje registrado. En las figuras 3.4 y 3.5 se puede observar la diferencia entre un MO dentro de los límites de la pantalla del ordenador (Figura 3.4) y un MO llevando la mirada hacia el extremo del ojo (Figura 3.5). A la hora de establecer el umbral, este debe ser lo suficientemente elevado como para evitar el ruido pero también lo suficientemente bajo como para detectar las señales. Si se tomase un umbral de amplitud dos (línea discontinua) los movimientos oculares de una amplitud más pequeña no serían detectados. Viendo estas imágenes, la solución inmediata a este problema de detección es simplemente bajar aún más el umbral, a una amplitud en la que no se llegue a detectar el ruido (amplitud uno, línea punteada).

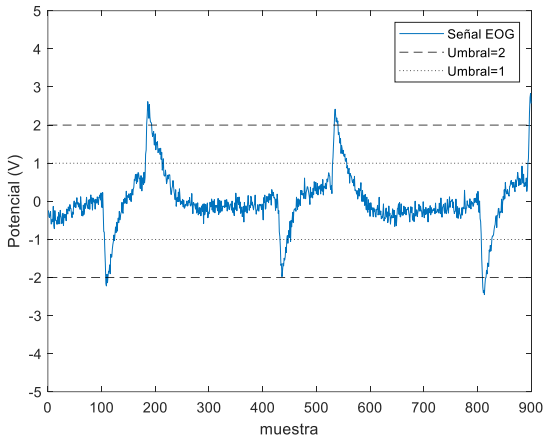

Figura 3.4 MO reducido

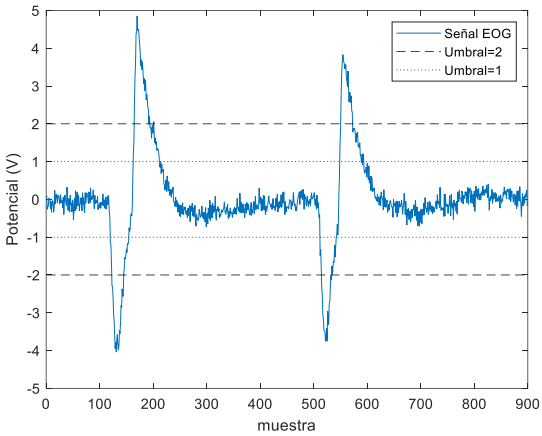

Figura 3.5 MO amplio

Sin embargo, cuando los entornos son más ruidosos se hace muy difícil o casi imposible diferenciar la señal del ruido cuando el movimiento ocular no es lo suficientemente amplio. Esto se ve claramente representado en la figura 3.6, que presenta una señal tomada con un par de cables y electrodos distintos a los de las figuras anteriores y muestreando esta vez movimientos hacia arriba y abajo. La señal representada en la figura 3.7 fue tomada segundos después con estos mismos electrodos pero esta vez realizando un movimiento ocular más amplio.

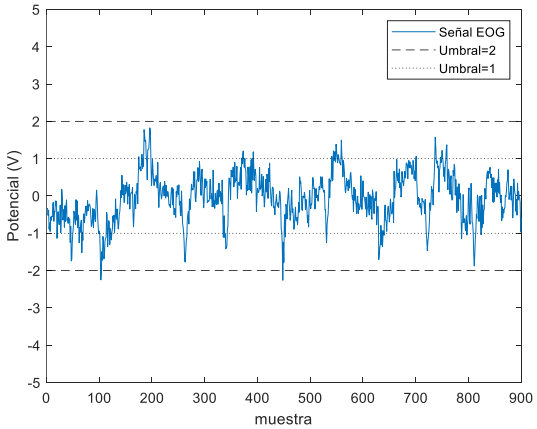

Figura 3.6 MO reducido con nivel de ruido elevado

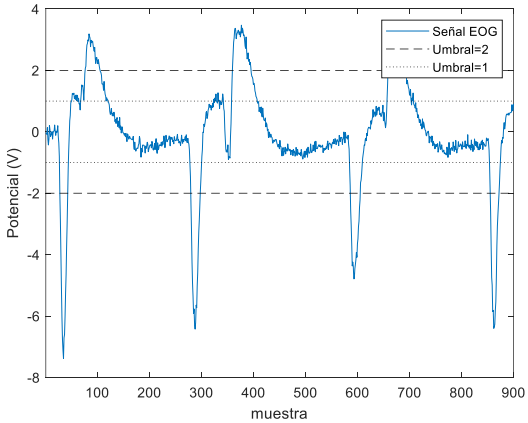

Figura 3.7 MO amplio con nivel de ruido elevado

Tras varias pruebas similares y en diferentes direcciones, se llegó a la conclusión de que la única forma de asegurar un nivel de señal a ruido lo suficientemente alto como para definir un umbral apto en la interfaz, era establecer como requisito para el usuario que los MO fuesen lo más amplios posibles.

De esta forma, las señales que servirán como comandos a la HCI se encuentran completamente definidas y diferenciadas del resto de posibles señales.

3.3. Implementación de la interfaz utilizando MATLAB

Una vez definidos los requisitos de la interfaz y la naturaleza de los movimientos a registrar, es momento de implementar la HCI en MATLAB. La señal es registrada y dividida en un array por canal en las funciones *Medir_mov_completo* y *prueba_mov_completo* respectivamente. La función que se debe implementar recibe como entrada un vector de muestras de la señal EOG cuyo tamaño depende del valor asignado a la variable *tambloque*.

Lo primero que debe hacer la función de MATLAB es convertir todos los datos del vector en '1', '0' o '-1' para luego poder procesarlos de forma más rápida. Para ello, se establece el umbral definido en el apartado 3.2.3 y se implementan las siguientes líneas de código:

```
1  %% Limpiar los datos del vector recibido
2  Vector_limpiado = zeros(size(vector_recibido));
3  for i = 1:length(vector_recibido)
4  if vector_recibido (i) >= limit
5  Vector_limpiado (i) = 1;
6  elseif vector_recibido (i) <= -limit
7  Vector_limpiado (i) = -1;
8  else
9  Vector_limpiado (i) = 0;
10 end
11 end
```

Código 3.1 Código para limpiar los datos del vector recibido

El valor de *limit* debe establecerse para cada usuario en función de la relación señal a ruido registrada en sus señales EOG. Tras esta limpieza se obtiene un tren de pulsos cuadrados que será analizado en el bucle principal de la función. En la figura 3.8 se representa el tren de pulsos correspondiente a la señal representada en la figura 3.7 con un umbral de amplitud dos.

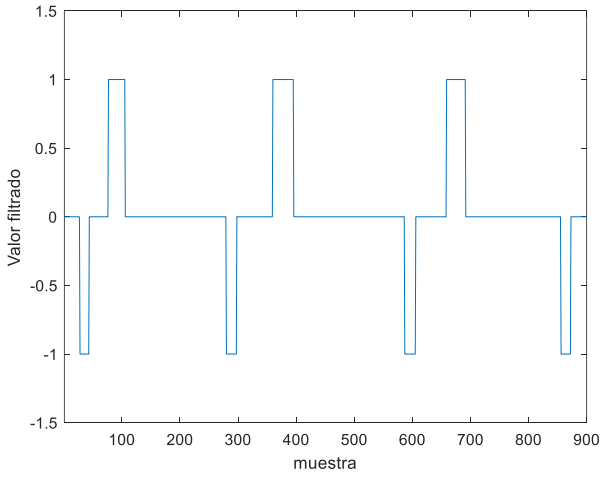

Figura 3.8 Vector filtrado

Como primer acercamiento, se propuso realizar una iteración a lo largo del vector para, mediante el uso de condiciones *if-else*, identificar cuándo se registraba un comando de movimiento a la derecha y cuándo uno a la izquierda. La figura 3.9 se introduce a modo de explicación sobre el funcionamiento de esta implementación. Tanto en esta figura como en las figuras 3.10 y 3.11, *cont_ceros* es una variable que almacena el número de '0' que se registran, *cont_neg* el número de '-1' y *cont_pos* el número de '1'.

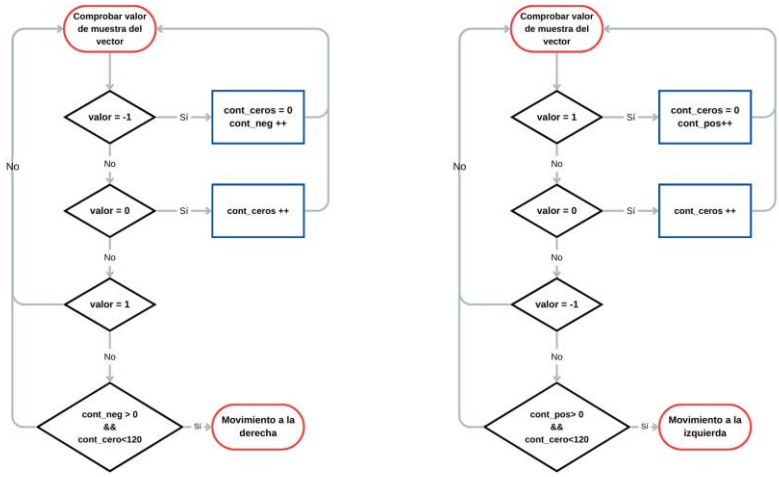

Figura 3.9 Diagrama de flujo de la primera versión del programa

Esta primera versión del programa requería que el vector limpiado fuese recorrido dos veces, una para detectar los movimientos a la derecha y otra para detectar los movimientos a la izquierda. Esto hacía que el tiempo de procesado fuese muy elevado por lo que se propuso una nueva implementación en la que se analizaba todo en el mismo bucle. El funcionamiento de esta implementación se muestra en la figura 3.10.

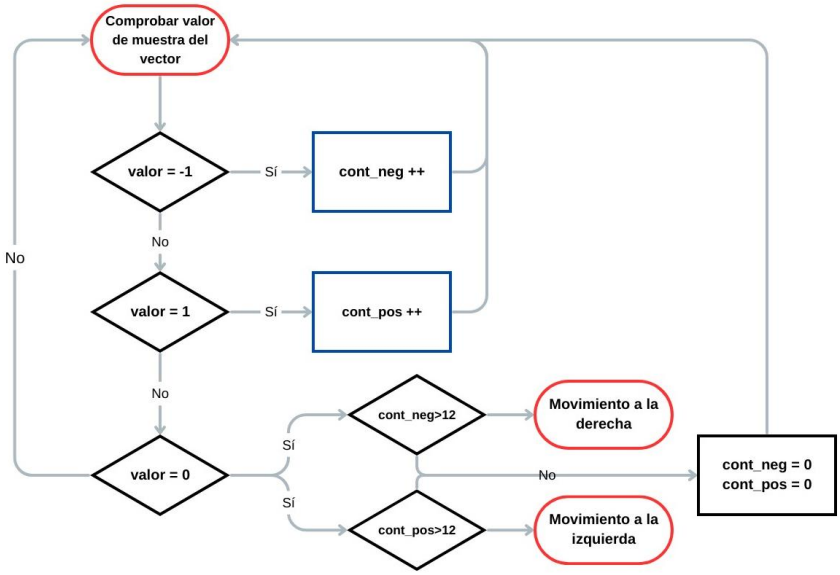

Figura 3.10 Diagrama de flujo de la segunda versión del programa

Esta nueva versión era más eficiente al solo necesitar que el vector fuese recorrido una vez. Sin embargo, a la hora de analizar las muestras, ocurría con frecuencia que los movimientos no eran detectados o eran identificados como movimientos en el sentido contrario. Para dar un enfoque más técnico y tras estudiar con más detenimiento las señales, se optó finalmente por implementar una máquina de estados. El diagrama de estados básico de esa máquina se puede observar en la Figura 3.11.

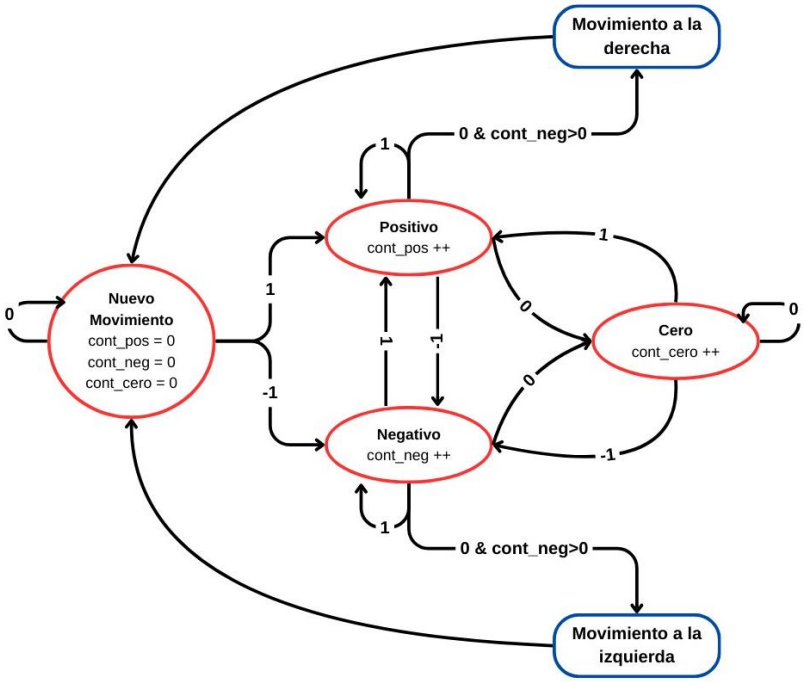

Figura 3.11 Máquina de estados a implementar

Teniendo en cuenta que las muestras que se reciben llegados a este punto ya han sido filtradas según el umbral definido y por tanto los valores de muestras posibles son ‘1’, ‘0’ y ‘-1’, el funcionamiento de la máquina de estados es el siguiente:

- La máquina comienza en el estado *Nuevo movimiento* y se mantiene en él mientras le lleguen muestras de valor ‘0’. Cada vez que se entra en este estado, todos los contadores se reinician a cero.
- Cuando se recibe una muestra de valor ‘1’, la máquina pasa al estado *Positivo*. En este estado se suma uno a un contador de positivos cada vez que aparece un ‘1’. Si por el contrario se recibe una muestra de valor ‘-1’ se pasa al estado *Negativo* y en él se suma uno a un contador de negativos cada vez que aparece un ‘-1’. Esto es aplicable no solo al estado *Nuevo movimiento* sino a todos los estados de la máquina y ayuda a identificar el ancho de los pulsos en el vector limpiado.
- Una vez la máquina se encuentra en estado *Positivo* o *Negativo* y recibe un ‘0’ hay dos opciones posibles:
  - Si está en el estado *Positivo* y el contador del negativo es cero o viceversa, se pasa al estado *Cero*.
  - Si está en el estado *Positivo* y ya se han contado algunos valores en el *Negativo* o viceversa, significa que se ha completado un comando de movimiento. Después, la máquina regresa al estado *Nuevo movimiento*.
- En el estado *Cero* se suma uno a un contador de ceros cada vez que llega un ‘0’ y se regresa a los estados *Positivo* o *Negativo* cuando se recibe un ‘1’ o un ‘-1’ respectivamente.

Un ejemplo de registro de un comando sería:

1. La máquina se encuentra en el estado *Nuevo movimiento*. Todos los contadores son inicializados a cero. Cuando recibe un '1' pasa al estado *Positivo*.
2. En el estado *Positivo* se reciben 20 muestras de valor '1' por lo que, el contador positivo tiene el valor 20 (*cont_pos = 20*).
3. Estando en el estado *Positivo* se recibe un '0', como el contador negativo vale 0 (*cont_neg = 0*) se pasa al estado *Cero*.
4. En el estado *Cero* se reciben 10 muestras de valor '0' por lo que el contador de ceros tiene el valor 10 (*cont_cero = 10*).
5. Estando en el estado *Cero* se recibe un '-1' por lo que se pasa al estado *Negativo*.
6. En el estado *Negativo* se reciben 20 muestras de valor '-1' por lo que el contador de negativos tiene el valor 20 (*cont_neg = 20*).
7. Estado en el estado *Negativo* se recibe un '0'. Como el contador positivo tiene un valor mayor a cero, esto significa que se ha recibido un pulso positivo seguido de una serie de ceros y un pulso negativo. En el apartado 3.2.1 del documento se especificó que esto correspondía a un movimiento a la izquierda por lo que, la máquina de estados lo interpreta como tal.
8. Finalmente se vuelve al estado *Nuevo movimiento* y se reinicializan todos los contadores a cero(*cont_pos = 0, cont_neg = 0, cont_cero = 0*).

Esta implementación está pensada únicamente para movimientos horizontales. La implementación para movimientos verticales sería similar cambiando derecha por arriba e izquierda por abajo.

En la implementación completa de la interfaz, se recibe un vector que contiene los valores de la señal EOG registrados en la tarjeta DAQ. Este vector contendrá los valores en solo uno de los ejes, vertical u horizontal. A continuación, la interfaz se encarga de limpiar este vector según el umbral definido para el usuario y posteriormente se implementa la máquina de estados que se ha definido. Cuando un comando de movimiento hacia una de las direcciones es detectado, la interfaz realiza las acciones asociadas a ese comando o llama a otra función que lo haga.

3.4. Pruebas y ajustes de la interfaz para garantizar su funcionamiento óptimo

Aunque la implementación propuesta en el apartado anterior funcionaría correctamente si todos los movimientos fuesen limpios y no hubiese ninguna interferencia, lo cierto es que el aspecto de los vectores limpiados es más parecido al que se presenta en la figura 3.13 que al de la figura 3.8. La figura 3.13 representa el resultado de filtrar la señal mostrada en la figura 3.12 con un umbral de amplitud uno. Los pulsos no siempre son perfectos y puede pasar que se fluctúe entre varios valores diferentes lo que, con el modelo definido anteriormente, sería interpretado por la interfaz como comandos.

En esta misma figura se representan tres casos en los que la interfaz anteriormente diseñada cometería errores a la hora de identificar el tipo de comando. A continuación, se analizarán estos casos y se describirá la solución propuesta en MATLAB para cada uno de ellos. De esta forma, se consigue que la interfaz tenga la eficiencia deseada.

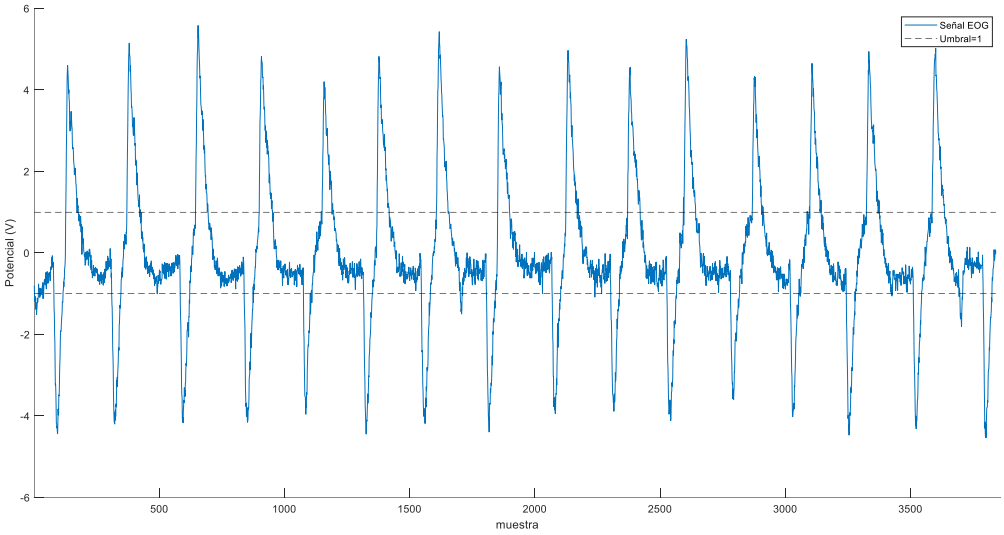

Figura 3.12 Señal EOG original

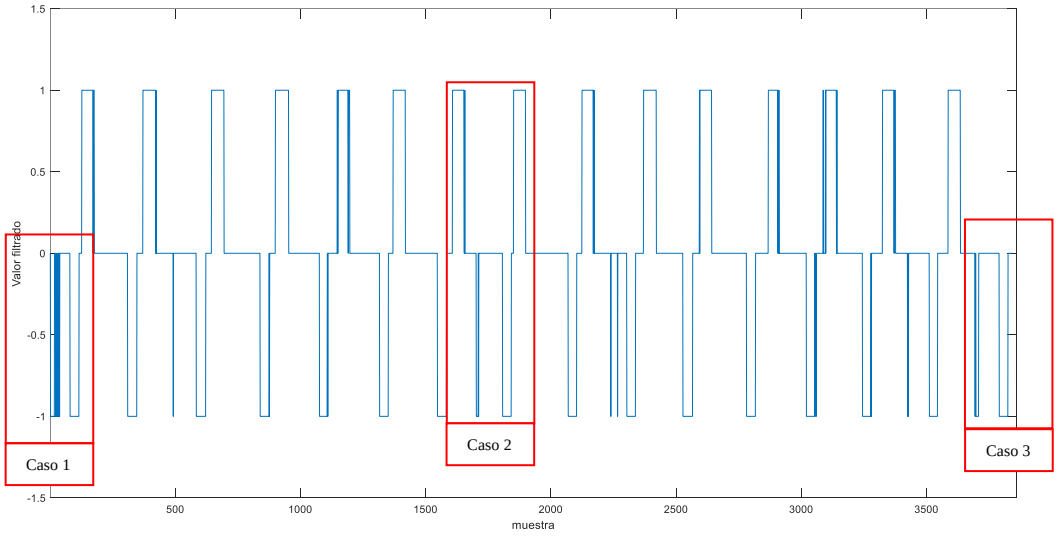

Figura 3.13 Señal EOG limpiada con umbral de amplitud 1.

Caso 1

Como se puede apreciar en la figura 3.13, la señal no siempre comienza en una serie de ceros seguido de una señal de comando sino que, muy a menudo, se producen una serie picos muy pequeños intercalados con ceros debido al alto nivel de ruido que presenta la señal en esa parte. Si, en lugar de como se da en este caso, la señal presentase a continuación un pico positivo indicador de iniciar un comando de movimiento hacia la izquierda, la interfaz lo identificaría como un comando de movimiento hacia la derecha (pico negativo seguido de positivo).

Para solucionar este problema, se optó por añadir un contador al estado inicial (*cont_inicial*). De esta forma, cada vez que se detecta un ‘1’ o un ‘-1’, en lugar de cambiar inmediatamente de estado, se incrementa en uno el valor de dicho contador. Cuando se haya

pasado por este estado inicial varias veces y se reciba un valor distinto de '0', entonces se cambiará de estado.

Es también importante asegurar que el valor anterior al que se ha recibido es el mismo que el que se recibe ahora. Es por esto por lo que se crea una variable llamada *valor_ant* que almacena el valor anterior para poder ser comprobado. La implementación en MATLAB de estas condiciones se puede leer subrayada en el código 3.2, que corresponde a un fragmento de código de la interfaz finalmente implementada.

```
14  case 'nuevo_mov'
15   % Acciones asociadas al estado
16   cont_pos = 0;
17   cont_neg = 0;
18   cont_cero = 0;
19   cont_inicial = cont_inicial+1;
20   % Transición al siguiente estado
21   if(valor == 1 && valor_ant==1 && cont_inicial>=5)
22    cont_inicial = 0;
23    estado_actual = 'positivo';
24   elseif(valor == -1 && valor_ant==-1 && cont_inicial>=5)
25    cont_inicial = 0;
26    estado_actual = 'negativo';
27   else
28    estado_actual = 'nuevo_mov';
29  end;
```

Código 3.2 Código del estado nuevo movimiento

Caso 2

A menudo se pueden observar en las señales picos que no corresponden realmente a ningún MO sino al efecto de un ruido mal filtrado. Para arreglar este problema se pueden añadir algunas condiciones más a la máquina de estados.

Cuando un pulso corresponde a una señal de pico, es más ancho que cuando corresponde a un poco de ruido. Para filtrar estos pequeños pulsos producidos por el ruido, se debe establecer un mínimo en el ancho de los pulsos para que sean considerados. Esto se traduce en añadir a la condición que permite identificar un movimiento a izquierda o derecha una condición de valor mínimo para los contadores de positivos y negativos.

```
(cont_neg>=min_ancho_pulso && cont_pos>=min_ancho_pulso)
```

Otra forma de evitar estos pulsos es añadir otra condición en los estados *Positivo* o *Negativo*. Cuando se reciben varios ceros seguidos y el pulso todavía no tiene el ancho requerido, se interpreta que lo recibido hasta el momento era ruido y que el movimiento aún no ha comenzado por lo que se debe regresar al estado inicial. En el código 3.3 se subrayan estas condiciones implementadas en el caso *positivo*. La implementación es análoga para el caso *negativo*.

```
33. case 'positivo'
34. % Acciones asociadas al estado
35. cont_pos=cont_pos+1;
36. % Transición al siguiente estado
37. if(valor == 1)
38.  estado_actual = 'positivo';
39. elseif(valor == -1)
40.  estado_actual = 'negativo';
```

```
41. elseif (cont_neg>=10 && cont_pos>=10)
42.  %%%MOVIMIENTO A LA DERECHA%%%
43.  % Transición al siguiente estado
44.  estado_actual = 'nuevo_mov';
45. elseif (cont_cero>=5 && valor_ant ~= 1)
46.  estado_actual = 'nuevo_mov';
47. else
48.  estado_actual = 'cero';
49. end;
```

Código 3.3 Código del estado positivo

Caso 3

A veces, se puede dar el caso de que un pulso de ruido justo después de un movimiento sea lo suficientemente ancho como para parecer un pulso de movimiento. En la figura 3.12 se observa cómo se produce un pulso de ruido algo más ancho que los demás justo antes del comienzo del último movimiento representado. Si tras este pequeño pulso se comenzase un movimiento a la izquierda (pulso positivo seguido de negativo) la interfaz podría interpretar que lo que se ha realizado es un movimiento a la derecha (pulso negativo seguido de positivo).

Para evitar esto, ya se decidió que el tiempo entre comandos debía ser lo suficientemente espaciado. En la interfaz, se debe configurar un número máximo de ceros que indique que ese bloque de ceros es la separación entre un movimiento y el siguiente. De esta forma, cuando el contador de ceros alcance este valor y vuelva a recibir un cero, se regresará al estado inicial, lo que indica que comienza un nuevo movimiento. El código correcto del estado Cero es el representado en la tabla de código 3.4.

```
69. case 'cero'
70. % Acciones asociadas al estado
71. cont_cero = cont_cero +1;
72. % Transición al siguiente estado
73. if(valor == 1)
74.  estado_actual = 'positivo';
75. elseif(valor == -1)
76.  estado_actual = 'negativo';
77. elseif (cont_cero>=150 && valor_ant == 0)
78.  estado_actual = 'nuevo_mov';
79. else
80.  estado_actual = 'cero';
81. end;
82.
```

Código 3.4 Código del estado cero

Con estas nuevas condiciones se consigue la fiabilidad de la interfaz aumente considerablemente. Lo que implica que la interfaz sea apta para su implementación en tiempo real.

3.5. Integración en tiempo real y retroalimentación visual para el usuario

Para integrar la interfaz en tiempo real se debe crear una función que se pueda integrar en la función *prueba_mov_total* antes mencionada. En este caso se deciden crear dos funciones separadas, una para cada canal (Vertical y Horizontal). Cada una de estas funciones recibe dos parámetros.

El primer parámetro es un vector de muestras de la señal EOG cuya longitud depende del tamaño de bloque definido al muestrear. Este tamaño también define cada cuanto tiempo se llama a la función *prueba_mov_total*. Cuanto menor sea el tamaño, más rápido será el procesado de los datos y antes se verán reflejadas las respuestas a los comandos en la interfaz. En este caso se ha definido un tamaño de 16 muestras por lo que se realiza una llamada a la función cada 125 milisegundos. En las pruebas realizadas a la interfaz se observó que el tiempo de procesado que requería para 16 muestras era de aproximadamente 80 milisegundos por lo que este tamaño de bloque es adecuado para el procesado de las señales en tiempo real.

El segundo parámetro es un puntero que se recibe de la función *Medir_mov_completo* mediante una variable global. En esta función se define un puntero que servirá posteriormente como retroalimentación visual para el usuario y para la evaluación de la interfaz. Consiste en un gráfico, mostrado en la figura 3.14, con un punto dibujado en el centro que se moverá en la dirección indicada mediante los comandos oculares previamente definidos. El *eje X* es el encargado de representar los movimientos horizontales y el *eje Y* los verticales.

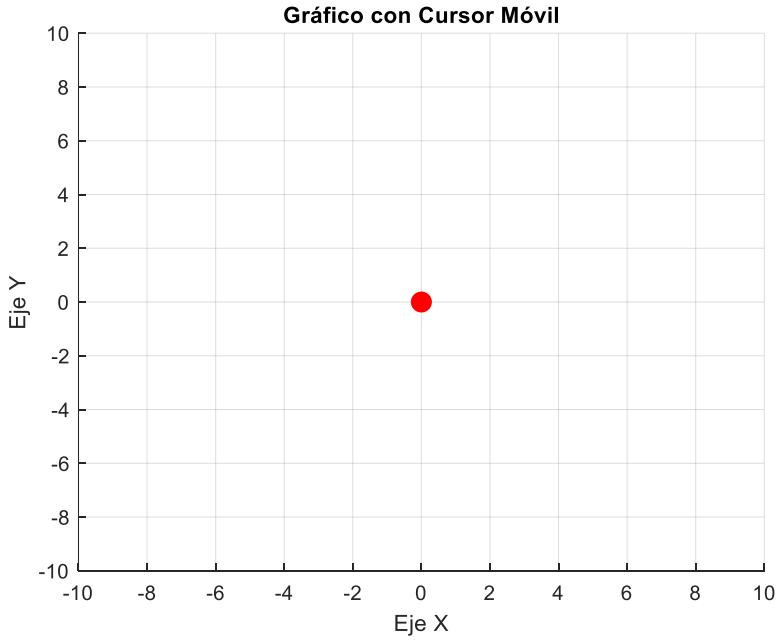

Figura 3.14 Representación gráfica del puntero

Es importante definir también una función que actualice el valor de este puntero cada vez que sea llamada. Es para esto que se ha definido la función *mover_cursor*. Esta función recibe como parámetros el puntero y el valor con el que debe ser desplazado en horizontal o en vertical.

Las funciones completas son *mov_horizontal* y *mov_vertical*. Es importante destacar que se han definido tres constantes que regulan: el ancho mínimo de los pulsos, el número de ceros que se deben contar para considerar que se va a realizar un nuevo movimiento y el valor del umbral para limpiar las señales recibidas. La existencia de estas constantes permite ajustar los valores de la interfaz a diferentes usuarios e instrumentos de medida que puedan presentar más o menos ruido.

Por otro lado, es necesario definir las variables que utiliza la máquina de estados como *persistent* para que no se reinicien en cada llamada a la función. Esto se debe a que la función

solo recibirá un número de muestras limitado en cada llamada y es necesario analizar un gran número de muestras para determinar que se ha producido un movimiento en alguna de las direcciones. La máquina de estados necesita conservar los valores tanto de estado actual como de los contadores. Estas variables se inicializan por primera vez solo cuando la función detecta que están vacías.

Tras la definición de las constantes y variables, solo queda limpiar el vector recibido e implementar la máquina de estados que realizará las funciones definidas en cada estado. Esta máquina será también la encargada de llamar a la función que moverá el cursor cuando identifique un comando de movimiento.

El usuario podrá identificar que el comando ha sido interpretado de forma correcta observando la nueva posición del cursor en el gráfico mostrado en pantalla.

Capítulo 4. Evaluación de la interfaz

4.1. Procedimiento de registro y análisis de datos de las pruebas

Para evaluar la efectividad de la aplicación se decidió hacer dos pruebas. Una para el plano vertical, y otra para el plano horizontal.

Las pruebas se realizan con un único sujeto debido a limitaciones en la disponibilidad de la tarjeta de adquisición de datos. El sujeto se sitúa en un entorno de laboratorio y se colocan los electrodos siguiendo el esquema representado en la figura 4.1. Cada prueba consta de un solo experimento que se explica con detalle a continuación.

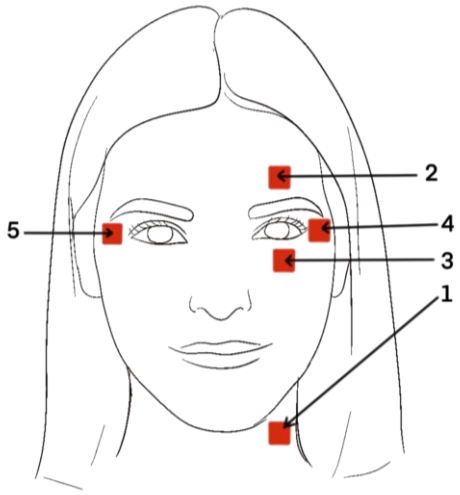

Figura 4.1 Esquema de colocación de electrodos.

1) Toma de tierra
2) Positivo canal Vertical
3) Negativo canal Vertical
4) Positivo canal Horizontal
5) Negativo canal Horizontal

Antes de comenzar los experimentos, se registra una serie de señales EOG que consisten en movimientos del usuario en todas las posibles direcciones. Esto sirve para poder definir las constantes de la interfaz asociadas al usuario. Estas constantes son: el ancho mínimo de los pulsos, el número de ceros que se deben contar para considerar que se va a realizar un nuevo movimiento y el valor del umbral para limpiar las señales recibidas. Además se debe definir el tamaño de bloque para el muestreo.

Los valores utilizados para ambos canales en este caso son los siguientes:

- Tamaño de bloque (*tam)*: 16
- Ancho mínimo de los pulsos (*min_ancho_pulso)*: 10
- Número de ceros entre comandos (*max_espacio_entre_movs*): 150
- Valor umbral de ruido (*limit*): 2

Primera prueba: movimientos en el plano horizontal

Para la primera prueba, se colocan únicamente los electrodos correspondientes al canal horizontal y a la toma de tierra (etiquetas 1, 4 y 5 de la figura 4.1).

El experimento realizado en esta prueba consiste en pedir al sujeto que realice una serie de comandos de movimiento horizontal. Para ello, se deben tener en cuenta todas las especificaciones sobre comandos mencionadas en capítulos anteriores. Un comando de movimiento a la derecha consiste en mirar lo más hacia la derecha posible y devolver la mirada al centro y un comando hacia la izquierda consiste en mirar lo más hacia la izquierda posible y devolver la mirada al centro. El sujeto debe comenzar el experimento fijando la mirada en el centro de la pantalla y dejar un tiempo de aproximadamente dos segundos entre un comando y el siguiente.

Para comprobar la efectividad de la interfaz en una situación real y no controlada, el sujeto debe alternar de forma aleatoria entre comandos hacia la derecha y comandos hacia la izquierda, realizando un total de 200 comandos.

Los comandos realizados por el usuario se verán reflejados en movimientos hacia la derecha y hacia la izquierda de un cursor representado en pantalla. Aunque el usuario puede comprobar en tiempo real si los comandos son interpretados correctamente, se decide grabar un vídeo para su análisis posterior. En el video se puede observar tanto al sujeto como a la pantalla del dispositivo en el que se ejecuta la interfaz.

Segunda prueba: movimientos en el plano vertical

Para esta segunda prueba se deben colocar solo los electrodos correspondientes al canal vertical y la toma de tierra (etiquetas 1, 2 y 3 de la figura 4.1).

El experimento realizado en esta prueba es casi igual al de la prueba anterior. La diferencia es que, en este caso, los comandos de movimientos son los correspondientes a arriba y abajo. Se debe recordar que un comando de movimiento hacia arriba consiste en dirigir la mirada al techo o el cielo y a continuación volver a mirar al centro de la pantalla, mientras que un comando de movimiento hacia abajo se realiza dirigiendo la mirada al suelo antes de volver a dirigirla al centro. Para este experimento también se debe comenzar fijando la mirada en el centro de la pantalla y se debe alternar entre comandos hacia arriba y hacia debajo de forma aleatoria, dejando el tiempo suficiente entre cada uno.

Se realizan de nuevo un total de 200 comandos que son grabados para su posterior análisis.

Otros experimentos

Además de los experimentos anteriormente mencionados, se pide al usuario que durante toda la prueba pestañee cuando lo sienta necesario así como que realice algún doble parpadeo de vez en cuando de forma aleatoria. Esto permite la evaluación de la respuesta de la máquina ante los pestañeos, que debería ser nula.

En principio, la interfaz debería ser capaz de identificar comandos de movimientos en diagonal, teniendo en cuenta los datos recibidos de ambos canales a la vez. Sin embargo, esta prueba no se pudo llevar a cabo debido a limitaciones en la disponibilidad del material de ensayo.

4.2. Resultados obtenidos y su interpretación

A continuación se detallan los resultados obtenidos en cada una de las pruebas realizadas al usuario.

Comando	Numero de comandos	Correcto	Incorrecto	Porcentaje de error
Izquierda	102	98	4	3,92 %
Derecha	98	95	3	3,06 %
Arriba	101	97	4	3,96 %
Abajo	99	89	10	10,10 %
Total	400	379	21	5,25%

Tabla 4.1 Resultados obtenidos

Movimientos a derecha e izquierda

De los 200 comandos aleatorios en ambas direcciones se realizaron un total de 98 comandos de movimiento a la derecha y 102 comandos de movimiento a la izquierda. Se obtuvo el resultado esperado en 95 de los 98 comandos de movimiento hacia la derecha y en 98 de los 102 comandos de movimiento hacia la izquierda. El resto de los comandos fueron interpretados erróneamente como movimientos en la dirección contraria.

En términos de estadística, solo un 3,5% de los comandos fueron interpretados erróneamente.

Movimientos arriba y abajo

De los 200 comandos aleatorios en ambas direcciones se realizaron un total de 101 comandos de movimiento hacia arriba y 99 comandos de movimiento hacia abajo. Se obtuvo el resultado esperado en 97 de los 101 comandos de movimiento hacia arriba y en 89 de los 99 comandos de movimiento hacia abajo. El resto de los comandos no fue detectado por la interfaz.

En términos de estadística, un 7% de los comandos no fueron detectados. Sin embargo, si separamos estos comandos según la dirección, vemos que la gran mayoría de los errores se dan cuando el usuario trata de mirar hacia abajo. Esto puede deberse a que es más difícil realizar movimientos suficientemente amplios cuando se dirige la mirada hacia abajo que en otras direcciones. Por lo que no se debería considerar esto como un fallo crítico de la interfaz.

Pestañeo

Tras realizar algunas pruebas aleatorias en las que el usuario pestañeaba, se detectó que, si los pestañeos eran frecuentes o muy seguidos unos de otros como un doble pestañeo, la interfaz interpretaba estos pestañeos como movimientos hacia arriba o abajo, de forma aleatoria. Sin

embargo, si se produce un pestañeo de vez en cuando, la interfaz es capaz de ignorarlo correctamente en la gran mayoría de los casos.

Resumen

La interfaz es capaz de interpretar los comandos realizados por el usuario de forma exitosa en un 94,75% de los casos. Este porcentaje es suficiente para considerar que la interfaz funciona de forma correcta y puede ser aplicable a futuros desarrollos en tiempo real.

Cabe destacar sin embargo, que las pruebas se han realizado a un solo usuario y en un entorno de laboratorio controlado, debido a las limitaciones en la disponibilidad de los instrumentos de medida durante el desarrollo de este TFG. Para concluir realmente que la interfaz funciona de forma correcta para cualquier usuario, se deberían realizar más pruebas a sujetos diferentes y en diferentes entornos.

Capítulo 5. Conclusión y líneas futuras

Durante el proyecto, se definieron y cumplieron varios objetivos específicos. Se investigaron las características de las señales EOG y sus aplicaciones en el control de interfaces, se diseñó e implementó una interfaz utilizando MATLAB y se realizaron pruebas exhaustivas para garantizar su funcionamiento óptimo. Los resultados obtenidos demuestran que es posible controlar de manera precisa un cursor en dos dimensiones a través de los movimientos oculares, logrando así una interacción natural y efectiva entre el usuario y el sistema.

Para futuros desarrollos se recomienda abordar, entre otros aspectos, una mejora en la precisión del sistema. Esta mejora se puede lograr a través de algoritmos de filtrado más avanzados y técnicas de aprendizaje automático para la clasificación de las señales EOG. Además, se podría ampliar la funcionalidad del sistema incorporando más comandos y funcionalidades como la detección de parpadeos. Esto podría hacer que la interfaz fuese más versátil y útil en diferentes contextos. Finalmente, la integración de sensores más avanzados y específicos para la captura de señales EOG podría mejorar la calidad de las señales registradas, reduciendo el ruido y aumentando la robustez del sistema.

La interfaz desarrollada tiene un gran potencial en aplicaciones médicas, especialmente para pacientes con discapacidades motoras. La adaptación de esta tecnología para controlar sillas de ruedas, dispositivos de comunicación o sistemas domóticos podría mejorar significativamente la calidad de vida de estos pacientes. En ese sentido, sería muy útil investigar la posibilidad de miniaturizar el sistema y desarrollar dispositivos portátiles o vestibles que permitan una mayor libertad de movimiento y comodidad para el usuario. Además, sería interesante explorar la integración de esta tecnología con otras tecnologías emergentes como la realidad aumentada o la realidad virtual, para crear experiencias de usuario más inmersivas y dinámicas.

En conclusión, este proyecto no solo ha demostrado la viabilidad de utilizar señales EOG para el control de dispositivos, sino que también ha sentado las bases para futuras investigaciones y desarrollos en el campo de las interfaces hombre-máquina. Con mejoras y expansiones, esta tecnología tiene el potencial de ofrecer soluciones innovadoras y accesibles para una amplia variedad de aplicaciones.

Referencias

[1] B. J. Adaptaciones, «El acceso a las TIC para personas con parálisis cerebral», El blog de Qinera. Accedido: 8 de mayo de 2024. Disponible en: https://blog.qinera.com/paralisis-cerebral-acceso-a-las-tic/

[2] N. M. Baquer, «Eye tracking: qué es, qué tipos hay y para qué sirve». Accedido: 8 de mayo de 2024. Disponible en: https://psicologiaymente.com/psicologia/eye-tracking

[3] «Tecnología de eye-tracking de IRISBOND. Integración», Irisbond. Accedido: 9 de mayo de 2024. Disponible en: https://www.irisbond.com/tecnologia-eye-tracking/

[4] skyvictor375, «Movimientos oculares», MÉTODOS PSICOFISIOLÓGICOS. Accedido: 11 de mayo de 2024. Disponible en: https://skyvictor375.wordpress.com/2015/04/13/movimientos-oculares/

[5] «Apple Vision Pro», Apple. Accedido: 13 de mayo de 2024. Disponible en: https://www.apple.com/apple-vision-pro/

[6] H. Mulam, M. Mudigonda, B. P. S. Kumar, y H. Kuchulakanti, «Electrooculogram Based Wheelchair Control in Real-Time», en *Proceedings of the Second International Conference on Emerging Trends in Engineering (ICETE 2023)*, vol. 223, B. Raj, S. Gill, C. A. G. Calderon, O. Cihan, P. Tukkaraja, S. Venkatesh, V. M. S., M. Mudigonda, M. Gaddam, y R. K. Dasari, Eds., en Advances in Engineering Research, vol. 223. , Dordrecht: Atlantis Press International BV, 2023, pp. 55-67. doi: 10.2991/978-94-6463-252-1_8.

[7] R. A, F. Shamim, S. Shams, M. Saleem, y R. Nisha, «EOG Based Text and Voice Controlled Remote Interpreter for Quadriplegic Patients», *VFAST Trans. Softw. Eng.*, vol. 12, n.º 1, Art. n.º 1, mar. 2024, doi: 10.21015/vtse.v12i1.1593.

[8] R. Zhang *et al.*, «An EOG-Based Human–Machine Interface to Control a Smart Home Environment for Patients With Severe Spinal Cord Injuries», *IEEE Trans. Biomed. Eng.*, vol. 66, n.º 1, pp. 89-100, ene. 2019, doi: 10.1109/TBME.2018.2834555.

[9] L. Gila, A. Villanueva, y R. Cabeza, «Fisiopatología y técnicas de registro de los movimientos oculares», *An. Sist. Sanit. Navar.*, vol. 32, pp. 9-26, 2009.

[10] «Anatomía del ojo humano», Cerebriti.com. Accedido: 14 de mayo de 2024. Disponible en: https://www.cerebriti.com/juegos-de-ciencias/anatomia-del-ojo-humano/

[11] «Novel Sensing Technology For Real Time Eye-Tracking Using Electrooculography (EOG) In Virtual And Augmented Reality (Wearable USA 2017)». Accedido: 16 de mayo de 2024. Disponible en: https://www.idtechex.com/en/event-presentation/novel-sensing-technology-for-real-

time-eye-tracking-using-electrooculography-eog-in-virtual-and-augmented-reality/10758
[12] «g.Bsamp». Accedido: 16 de mayo de 2024. Disponible en: https://hooshmandfanavar.com/en/g-tec-medical-engineering-solutions/g-bsamp/
[13] «Model». Accedido: 16 de mayo de 2024. Disponible en: https://www.ni.com/es-es/shop/model/usb-6210.html