DEPARTAMENTO DE MATEMÁTICAS APLICADAS Y SISTEMAS

CONTROL DE UN ROBOT UTILIZANDO SEÑALES DE MOVIMIENTOS OCULARES.

PROYECTO TERMINAL

Que para obtener el grado de

INGENIERO EN COMPUTACIÓN

presenta

KEVIN ENRIQUE OLVERA ORTEGA

Asesores:

Dra. Alicia Montserrat Alvarado González &
Dr. Antonio López Jaimes

Ciudad de México, MÉXICO

KEVIN ENRIQUE OLVERA ORTEGA: Control de un robot utilizando señales de movimientos oculares.. Ingeniero en Computación. ©

SUPERVISOR: Dra. Alicia Montserrat Alvarado González &
Dr. Antonio López Jaimes

CONTACTO: kevinenriqueoogm@gmail.com

AGRADECIMIENTOS

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity,
and especially because it produces objects of beauty.

— Donald Ervin Knuth

Para mi mamá, todos estos son tus logros madre mía.

ÍNDICE GENERAL

1 INTRODUCCIÓN

Una Interfaz Cerebro-Computadora (BCI, *Brain Computer-Interface*) es una tecnología diseñada para comunicar a un usuario con una computadora usando únicamente señales cerebrales Kübler, 2000. Este tipo de tecnología permite a las personas extender sus capacidades al dotarlas de habilidad para mandar instrucciones a un dispositivo físico, sin tener que usar ninguna clase de actividad corporal generada por el sistema nervioso periférico. Tan solo requieren sus capacidades cognitivas, transformando así, los pensamientos en acciones reales sobre el ambiente.

Las BCI pueden ayudar a las personas con problemas de movilidad, ya sea por consecuencia de daños producidos por accidentes o por enfermedades. Por ejemplo, en el caso de la Esclerosis Lateral Amiotrófica (ELA, la cuál padecía el famoso científico británico Stephen Hawking), se caracteriza por una pérdida gradual de las neuronas motoras en el cerebro y la médula espinal Orient-López, 2006. Un sistema de BCI permitiría al individuo reemplazar sus propias habilidades motoras con instrucciones que pueda mandar a un robot o incluso una prótesis o un brazo robótico.

Las BCI se apoyan de tecnologías ya existentes para medir la actividad cerebral. Dentro del campo de las neurociencias, la lectura de señales de electroencefalograma (EEG) no es el único método que se tiene para medir esta actividad. Existen una gran variedad de métodos, e.g., las Imágenes de Resonancia Magnética funcional (fMRI), la Tomografía por Emisión de Positrones (PET), o la Espectroscopia Funcional del Infrarrojo Cercano (fNIRS) González, 2016. Cada método presenta ventajas y desventajas, no obstante, dos parámetros de comparación muy usados en la literatura son la resolución espacial y la temporal. La primera define la cantidad de neuronas que pueden ser analizadas dentro de un área determinada, y la segunda el tiempo que lleva el tomar una nueva muestra de datos. Métodos como PET y fMRI no son adecuados para BCI porque no tienen una buena resolución temporal, que es necesaria para aplicaciones en tiempo real, además, requieren de una infraestructura muy costosa. En cambio, la señal de EEG ofrece información muy cercana a la generación del estímulo, lo que la hace adecuada para nuestro sistema Kıymık y col., 2005. A pesar de que su resolución espacial no es la mejor, es suficiente

para los fines de nuestro sistema, además de ser considerablemente más barata que otras tecnologías.

1.1 MOTIVACIÓN Y CONTEXTO DEL PROBLEMA

Mientras que la motivación para hacer una BCI que controle una prótesis es evidente, las aplicaciones que puede tener controlar un robot mediante señales cerebrales pueden ser menos obvias. Una primera aplicación puede venir por parte de personas con movilidad reducida que quieran usar al robot como un asistente personal capaz de transportar objetos hasta ellos o incluso realizar labores de limpieza. El ejemplo perfecto de una persona que necesitaría de estos servicios es un enfermo en estado de postración.

Otra motivación para el desarrollo de este tipo de sistemas es permitir una interacción más flexible con los sistemas de cómputo, dado que el control de dispositivos electrónicos se encuentra restringido al uso de *joysticks* y controles tradicionales como el *mouse*, *trackpad*, teclado y panel táctil. Suresh y Schwager, 2016. Una nueva forma de interacción permitiría más libertad al usuario para manejar su propio cuerpo como mejor le convenga e incluso eliminaría la necesidad de adoptar posiciones poco ergonómicas.

Los sistemas BCI también pueden ser utilizados en la industria militar, donde los cuerpos de seguridad pueden ocuparlos para tareas de rescate, primeros auxilios y defensa Munyon, 2018.

1.2 ANTECEDENTES

En 1924, Hans Berger (psiquiatra alemán, Figura 1) fue la primera persona en grabar la actividad cerebral mediante EEG. Este método recoge la actividad eléctrica del cerebro usando electrodos colocados sobre el cuero cabelludo Kalagi y col., 2017.

En los años 70's investigaciones conjuntas entre la UCLA y la *National Science Fundation*, fueron las primeras en introducir el término *Brain-Computer Interface* Kübler, 2020.

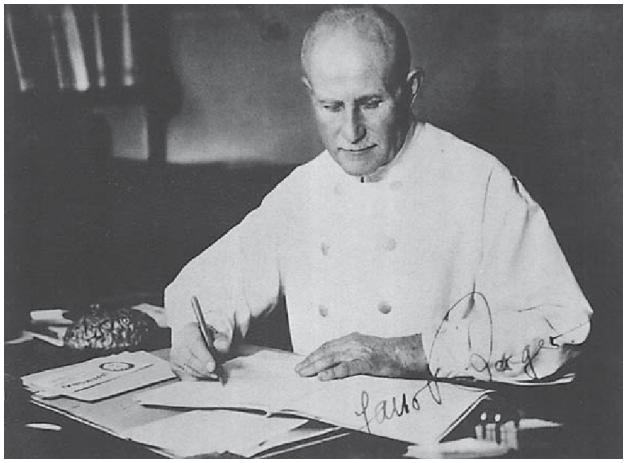

Figura 1: Hans Berger, año desconocido. Imagen perteneciente al Departamento de Neurología y Rehabilitación de la Universidad de Illinois en Chicago, Chicago, Illinois, EE. UU.

1.3 PLANTEAMIENTO DEL PROBLEMA

El problema que este proyecto busca resolver es la comunicación entre un robot y el cerebro del usuario. El propósito de la interfaz es permitir controlar el movimiento del robot mediante señales de EEG. Las señales son capturadas por un casco de EEG al tiempo que se procesan usando diversos algoritmos para filtrar el ruido y decodificar las intenciones del usuario para transformarlas en instrucciones ejecutables por el robot.

1.4 DESCRIPCIÓN DEL SISTEMA

El sistema (Figura 2) comienza desde la adquisición de los datos con el casco de EEG, donde los electrodos recogen directamente del cuero cabelludo del sujeto la señal a analizar. Esta señal es enviada hasta la tarjeta Cyton de manera automática, la cual a su vez está conectada a un puerto USB de la computadora encargada de procesar los datos O. D. Team, 2021a. Una vez que los datos ya están en la computadora, estos son recogidos y analizados por los algoritmos. Un programa escrito en Python es el que realiza esta función. El programa aplica los filtros correspondientes y detecta los picos en la señal que son generados por el movimiento ocular para transformarlos en instrucciones

tales como avanzar o girar. Estas instrucciones son enviadas al controlador del robot en ROS. Este controlador funciona como un *middleware* o agente intermedio entre el software del ordenador y el hardware del robot. Es decir, las instrucciones que son enviadas al robot no son instrucciones en bajo nivel para el hardware del robot, sino que están escritas en un lenguaje intermedio que puede ser fácilmente manejado por los desarrolladores para que construyan aplicaciones que usen el robot.

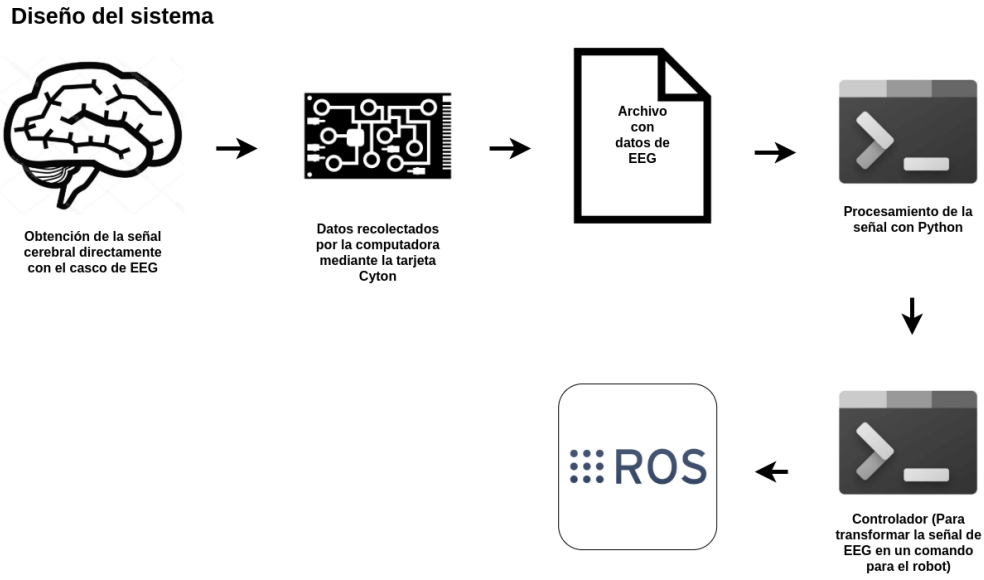

Figura 2: Diseño del sistema. Se muestran a manera de módulos los componentes lógicos y físicos que hacen posible convertir las instrucciones detectadas desde el casco hasta la simulación en ROS.

1.5 OBJETIVOS

Objetivo General

Desarrollar un algoritmo que detecte movimientos oculares en el EEG que permitan manipular un robot virtual.

Objetivos Particulares

1. Leer en tiempo real las señales del EEG y almacenarlas en un archivo de texto.
2. Preprocesar las señales de EEG para eliminar ruido.
3. Implementar un algoritmo para detectar las señales generadas por movimientos oculares.
4. Probar el algoritmo con señales de un conjunto de datos de referencia.
5. Utilizar las señales de los movimientos oculares para manipular un robot virtual.

2 CONCEPTOS BÁSICOS

2.1 TRABAJO RELACIONADO

Este proyecto esta basado fundamentalmente en el trabajo del artículo "*Brain-Swarm Interface (BSI): Controlling a Swarm of Robots with Brain and Eye Signals from an EEG Headset*", escrito por Aamodh Suresh de la Universidad de California, San Diego Suresh y Schwager, 2016. En este se realiza una interfaz capaz de controlar el movimiento y tamaño de un enjambre de robots utilizando como parámetros los pensamientos y los movimientos oculares. Tanto los pensamientos del usuario como los movimientos oculares se obtienen mediante un encefalograma. Para la detección del movimiento ocular se utilizan técnicas de filtrado de ruido, mientras que, para detectar los pensamientos del usuario se utiliza una técnica llamada Modelo de Markov Oculto. Para nuestro proyecto se utilizan solamente las señales provenientes del movimiento ocular y está diseñado para controlar un solo robot.

Muchos otros trabajos se han realizado utilizando interfaces cerebro computadora algunos de ellos, tal como el nuestro, tienen como propósito el controlar un movimiento mecánico usando señales cerebrales. En la mayoría de los casos, se considera que los parámetros cinemáticos para mover un elemento sobre dos dimensiones solo pueden ser calculados con precisión si se usan microelectrodos intracórticales (situados al interior de la corteza cerebral). El trabajo realizado por Gerwin Schak titulado "*Decoding two-dimensional movement trajectories using electrocorticographic signals in humans*" es parecido al nuestro Schalk, Kubanek y col., 2007. Su propósito es transformar las señales recogidas en movimiento mecánico sobre un plano bidimensional, no obstante, este trabajo utiliza electrodos colocados entre la superficie del cerebro y la capa exterior que lo cubre y protege (duramadre), por el contrario, nuestro trabajo utiliza un método de recolección de señales no invasivo al recoger las señales con casco con electrodos que se coloca sobre el cuero cabelludo.

Otros casos, como el del trabajo "*BCI2000: a general-purpose brain-computer interface (BCI) system*", nuevamente escrito por Gerwin Schalk, han avanzado más en el desarrollo de este tipo de sistemas Schalk, McFarland y col., 2004. En este trabajo se crea una interfaz cerebro-computadora de propósito general que

permite combinar diferentes métodos de obtención de las señales cerebrales, con diferentes algoritmos de procesamiento y protocolos de operación. Trabajos de este tipo son de gran utilidad para toda la comunidad científica, pues permiten descomponer en módulos las diferentes partes de los sistemas construidos hasta el momento, de esta manera se pueden trabajar por separado los diferentes módulos de los sistemas de BCI, y por ejemplo modificar solamente el algoritmo de operación para adecuarlo a un proyecto específico.

2.2 SEÑALES EEG

Es una técnica que permite estudiar la actividad cerebral, es el registro y evaluación de los potenciales eléctricos generados por el cerebro y obtenidos por medio de electrodos situados sobre la superficie del cuero cabelludo Barreto Galeano, 2014. Las neuronas se comunican mediante impulsos eléctricos y están activas en todo momento. Esta actividad eléctrica es recogida por el electroencefalograma. La señal ofrecida por el EEG es temporalmente muy cercana a la generación del estimulo, a procesos psicológicos y a la respuesta cortical observada. Esto nos ofrece la posibilidad de analizar la información en tiempo real y con una precisión temporal de milisegundos González, 2016.

Un EEG permite obtener información del cerebro mientras este realiza actividades de manera consciente e inconsciente. La actividad cerebral que interesa para controlar los dispositivos de BCI es aquella que puede ser provocada y controlada por el sujeto de manera consciente (actividad asincrónica) ya que siempre se busca que refleje un acto de voluntad por parte del sujeto para transmitir un mensaje González, 2016.

Los riesgos del EEG

Los electroencefalogramas han sido usados por muchos años y se consideran un procedimiento seguro ya que no causan incomodidad, los electrodos simplemente graban la actividad cerebral y no producen ninguna sensación. Puede ocurrir irritación y enrojecimiento de la piel en la zona donde se colocaron los electrodos, pero esta desaparecerá después de unas cuantas horas.

2.3 EL SISTEMA DE ADQUISICIÓN

El sistema de adquisición de EEG usado en este proyecto consta de un casco y una tarjeta de biosensado que se comunica directamente con una computadora.

El casco utiliza electrodos superficiales e indoloros que están distribuidos de acuerdo con el sistema internacional de electrodos 10-20 Morley y col., 2016. Este sistema es un estándar para la colocación de los electrodos y posibilita la reproducción de los experimentos y de los estudios. Su nombre se debe a que los electrodos están espaciados entre 10 % y 20 % de la distancia total entre puntos reconocibles de cráneo. Estos puntos son el Nasión (puente de la nariz), el Inión (protuberancia en la parte posterior de la cabeza arriba de la nuca) y los puntos preauriculares (delante del trago de cada pabellón de la oreja). La Figura 3 muestra un diagrama de la localización de los electrodos en este sistema.

En el sistema 10-20, cada sitio tiene una letra para identificar el lóbulo y un número para identificar la ubicación del hemisferio. Así, F se usa para identificar el lóbulo Frontal, T el temporal, C el central, P el parietal y O el occipital. Los números 2, 4, 6 y 8 corresponden a los electrodos colocados en el hemisferio derecho y 1, 3, 5, 7, y 9 a los del hemisferio izquierdo. Los electrodos en la línea media no son señalados mediante un número, en su lugar se utiliza una z Morley y col., 2016.

Un electrodo es un conductor eléctrico diseñado para hacer contacto con una parte no metálica de un circuito Faraday, 1834. Generalmente se implementa usando un disco de metal que puede estar hecho de cloruro de plata o de estaño. Forma una conexión eléctrica con el cuero cabelludo (pudiendo o no usar un gel conductor). Adicionalmente, el circuito formado por la piel, el gel y el electrodo puede funcionar como un capacitar que atenúa la transmisión de las frecuencias bajas. Por lo que es importante asegurar las mejores condiciones de adquisición de la actividad eléctrica.

Por otro lado, se utiliza la tarjeta de biosensado Cyton OpenBCI. Esta es compatible con Arduino. Se encarga de recolectar la señal análoga recibida directamente por los electrodos y transformarla en una señal digital que pueda ser enviada a una computadora para su procesamiento. Particularmente la tarjeta de biosensado Cyton utilizada para este proyecto cuenta con una interfaz neuronal de 8 canales, extensible a 16 cuando se le agrega el módulo Daisy (*OpenBCI Daisy Module*). Cuenta con un procesador de 32 bits. Esta tarjeta además de tomar muestras de la actividad cerebral (EEG), también puede tomar

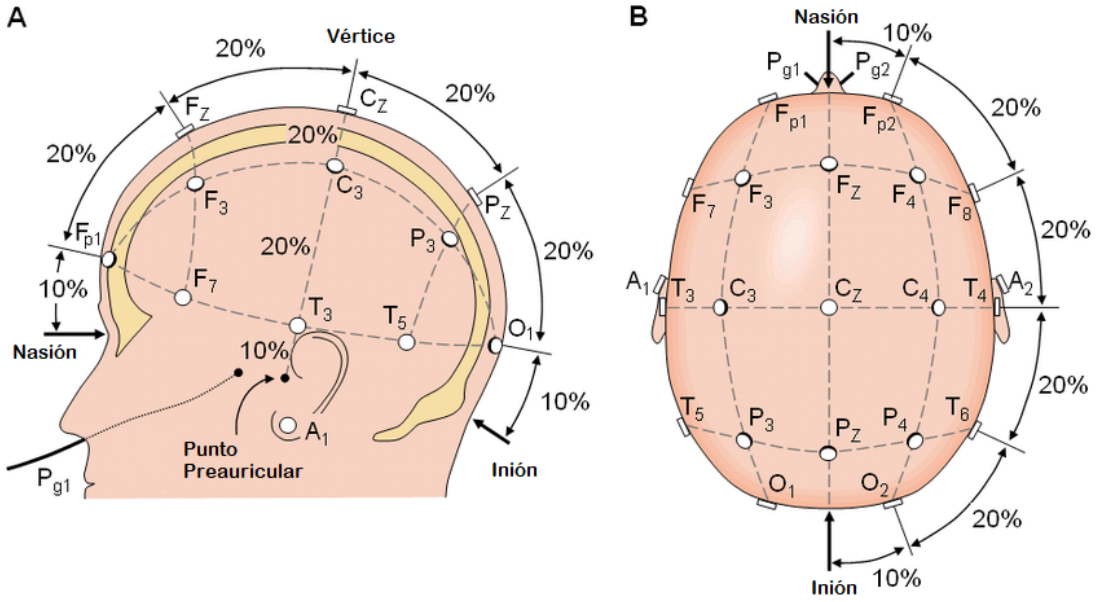

Figura 3: Sistema internacional de colocación de electrodos 10-20. En el lado izquierdo de la imagen se puede apreciar una vista lateral de una cabeza humana, sobre ella están distribuidos los electrodos del casco separados entre ellos 20 % de la distancia total desde Nasión hasta el Inión. Los electrodos F_{p1} y F_{p2} están ambos separados un 10 % separados del Nasión cada uno y 20 % separados entre ellos como se puede apreciar del lado derecho. Tomada de "Mapeo electroencefalográfico y Neurofeedback", por Novo-Olivas y col., 2010.

muestras de datos de actividad muscular y cardiaca. Se comunica de manera inalámbrica a una computadora mediante la OpenBCI USB usando módulos de RF Arduino. También puede comunicarse de manera inalámbrica a cualquier dispositivo compatible con Bluetooth de baja energía (BLE) O. D. Team, 2021a.

Por defecto la tarjeta Cyton envía información de los canales 60 veces por segundo, es decir, trabaja a una frecuencia de 60 Hz. Pero es posible aumentar esta frecuencia a un máximo de 250 Hz usando 8 canales. Si se usan 16 canales la frecuencia máxima es solo de 125 Hz.

2.4 SEÑALES EOG

Un EEG mide las señales eléctricas generadas por el cerebro, dentro de estas señales viene ruido eléctrico generado por el movimiento ocular y demás actividad muscular.

Para este trabajo se aprovecha el concepto de BCI Híbrida popularmente conocida como hBCI. Esta puede utilizar información de cualquier otro sistema además del sistema BCI Pfurtscheller y col., 2010. En nuestro caso se utilizan las señales residuales del EEG generadas por el movimiento ocular que son comúnmente desechadas por los sistemas BCI. Específicamente, la señal ocular es usada como el control con el que el usuario determina el movimiento del robot.

Un electroculograma mide los potenciales eléctricos de la piel cercana a los ojos. El electrooculograma es capaz de medir el cambio en el potencial eléctrico entre la cornea y el fondo del ojo durante periodos sucesivos de adaptación a la luz y oscuridad. Brown y col., 2006.

En nuestro trabajo decidimos usar los electrodos F7 y F8 del Sistema internacional 10-20 de colocación de electrodos ya que son las posiciones más cercanas que se tienen dentro de este sistema a los ojos derecho e izquierdo. Esto nos permite observar dentro de la señal que recibimos del casco, los eventos relacionados al movimiento horizontal de los ojos (Figura 4). Aunque estas posiciones de los electrodos no son tan precisas como lo serían en un electrooculograma, aportan la suficiente información del movimiento ocular sobre el plano horizontal que nuestro proyecto necesita.

2.5 PROCESAMIENTO DE LOS DATOS

Los datos que llegan del casco a la computadora representan los voltajes adquiridos por los electrodos. Así, cada canal de información de cada electrodo llega a la tarjeta Cyton Daisy como una señal eléctrica analógica. La tarjeta a su vez amplifica esta señal y la convierte en una señal digital o discreta, es decir, una secuencia de valores que representan las mediciones de cada electrodo en un tiempo determinado.

Para procesar la señal, el primer paso es centrar la señal usando las Ecuaciones 1 y 2. La primera sirve para tomar el promedio de la señal de un canal usando una muestra significativa (mínimo 640 muestras de ese canal) y la segunda para restar a la muestra actual el promedio calculado. Sea u_i el canal

asociado al electrodo i, $u_i\prime$ la señal centrada, $\overline{u_i}$ el promedio del canal u_i y N el tamaño de la muestra para el promedio. La Ecuación 2 describe el algoritmo que centra la señal de ese canal. El segundo paso es dividir los datos de los canales en ventanas, cada ventana es en principio consecutiva a la anterior y no hay traslapes. A cada ventana se le aplica un filtro tipo Butterworth pasa bajas (ver Sección 2.6) Ellis, 2012; Suresh y Schwager, 2016. Posteriormente se siguen los pasos descritos en el Algoritmo 1 para detectar los picos en las señales que corresponden con el movimiento ocular sobre el eje horizontal.

$$\overline{u_i} = \frac{\sum_{n=1}^{N} u_i(t)}{N} \tag{1}$$

$$u_i\prime(t) = u_i(t) - \overline{u_i} \tag{2}$$

2.6 FILTROS DIGITALES

En procesamiento de señales, un filtro tiene cómo entrada una señal y cómo salida otra que pudo haber cambiado en **amplitud**, **periodo** o **fase**. El primer concepto es la distancia entre el punto más alejado de una onda y su punto de equilibrio. El segundo es el lapso mínimo que separa dos instantes en los que el sistema se encuentra en el mismo estado: mismas posiciones, mismas velocidades, mismas amplitudes, en otras palabras, es el tiempo que dura un ciclo de la onda en volver a comenzar (Estos dos primeros conceptos pueden observarse en la Figura 5. Mientras que el tercero es la fracción del periodo transcurrido desde el instante correspondiente al estado tomado como referencia. Si representáramos un ciclo como los 360° de un circulo, una diferencia en fase sería un diferencia en grados entre un punto sobre este circulo y el punto tomado como referencia. Consecuentemente dos señales que tienen la misma forma pueden estar desacopladas si tienen distinta fase (Figura 6). Ballou, 2013; *Book: Celestial Mechanics (Tatum)* 2021. La **frecuencia** es el número de veces que se repite una onda por unidad de tiempo Girón y de Córdoba. Departamento de Fısica Aplicada, 1992. En la Figura 7 puede observarse la variación en la frecuencia de una onda.

Comúnmente en procesamiento de señales los filtros se usan para quitar características y componentes no deseados en una señal. Generalmente para lograr este propósito se usan **filtros lineales** (su nombre se debe a que aplican un operador o función lineal a una señal para transformarla), que son aquellos diseñados para eliminar ciertas frecuencias y permitir el paso de otras. Una for-

ma de clasificar este tipo de filtros es por el rango de frecuencias que permiten pasar, de manera general se reconocen cuatro. Hamming, 1998. Estos son:

- **Filtros pasa-baja**: Este diseño de filtro se caracteriza por dejar pasar frecuencias bajas en su banda de rechazo determinadas por una frecuencia de paso FRP. Es decir, la banda de paso va en el rango de frecuencia $[0, FRP]$. Cualquier frecuencia por encima de FRP es rechazada.

- **Filtros pasa-alta**: Este diseño de filtro se caracteriza por dejar pasar frecuencias altas en su banda de rechazo determinadas por una frecuencia de paso FRP. Es decir, cualquier frecuencia por encima de FRP es aceptada y toda señal inferior a FRP es rechazada.

- **Filtros rechaza-banda**: Este tipo de filtros tienen dos frecuencias de paso FRP_a y FRP_b. Los filtros de este tipo rechazan las frecuencias situadas dentro del rango $[FRP_a, FRP_b]$ y permiten el paso de cualquier otra frecuencia que esté fuera de él.

- **Filtros pasa-banda**: A diferencia de los filtros rechaza-banda, las frecuencias situadas entre $[FRP_a, FRP_b]$ son permitidas y, las que están fuera, rechazadas.

La metodología moderna para el diseño de filtros lineales es llamada **síntesis de redes**. Una de las familias de filtros que se logran usando esta técnica son los filtros de Butterworth. Este tipo de filtros están diseñados para tener una respuesta de frecuencia (en inglés *frequency response*, se trata de la medida cuantitativa del espectro de salida en respuesta aun estimulo Smith, 2007) tan plana como sea posible en el rango de frecuencias de paso Bianchi y Sorrentino, 2007. En este proyecto se aplica un filtro Butterworth pasa bajas, esto nos permite aplanar las crestas y valles mas largos de la señal para que esta sea más parecida a una linea recta que nos permita distinguir bien las señales provenientes del movimiento ocular.

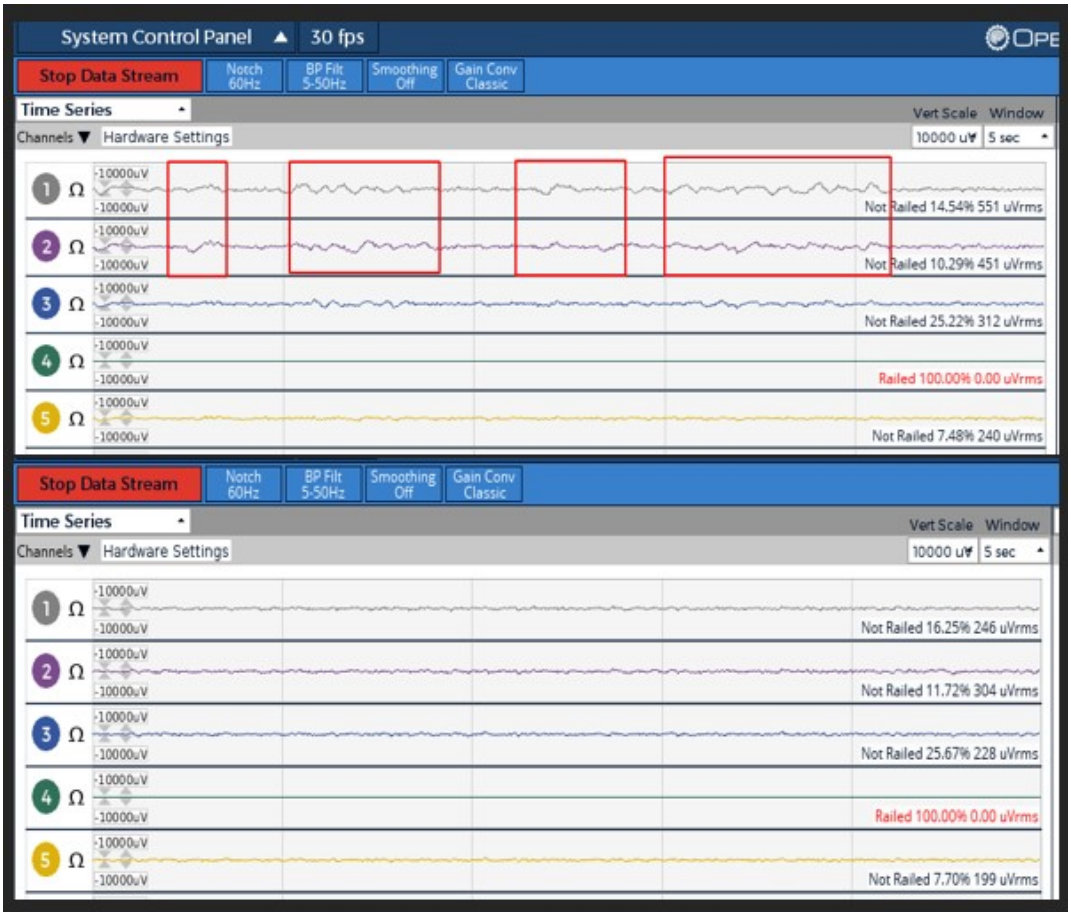

Figura 4: Esta imagen corresponde a dos capturas de pantalla del programa OpenBCI-GUI (que nos permite visualizar de manera gráfica los datos que la computadora recibe de la tarjeta Cyton Daisy). En la parte superior se muestran marcados en rojo sobre los canales 1 y 2 (correspondientes a los electrodos F7 y F8 respectivamente) los picos y valles que se producen en las señales cuando se mueven los ojos de izquierda a derecha. En la parte inferior las señales de estos canales lucen más planas pues el usuario permanece en reposo sin mover los ojos.

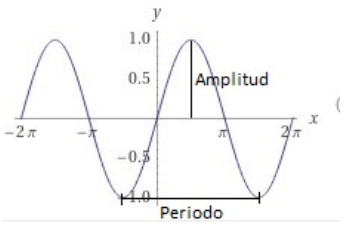

Figura 5: Representación gráfica de la Amplitud y Periodo en la señal producida por la función $\sin(x)$.

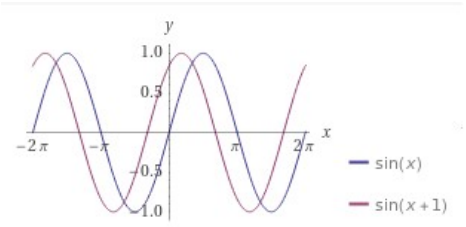

Figura 6: Las funciones $\sin(x)$ y $\sin(x+1)$ producen la misma señal en diferente fase.

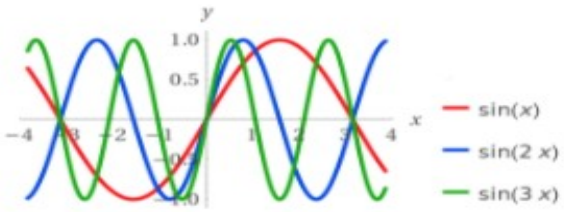

Figura 7: Las ondas dibujadas representan a la función $\sin(x)$, $\sin(2x)$ y $\sin(3x)$. La señal con una frecuencia más alta es $\sin(3x)$ pues tiene más ciclos por unidad en el eje X, mientras que la señal de más baja frecuencia es $\sin(x)$.

3 METODOLOGÍA

3.1 EL ALGORITMO PARA LA DETECCIÓN DEL MOVIMIENTO OCULAR HORIZONTAL

El algoritmo usado en este proyecto es tomado del artículo *"Brain-Swarm Interface (BSI): Controlling a Swarm of Robots with Brain and Eye Signals from an EEG Headset."* Escrito por Aamodh Suresh de la Universidad de California, San Diego. Suresh y Schwager, 2016.

A grandes rasgos, se realiza la detección del movimiento ocular usando electrodos separados entre sí y cerca cada uno de uno de los ojos del portador del casco de EEG. Dos electrodos se usan para detectar el movimiento horizontal y otros dos para detectar el movimiento vertical.

Este proyecto implementa el Algoritmo 1, usado para calcular solamente el movimiento horizontal usando los electrodos F7 y F8. Este algoritmo consta de cinco pasos principales que son:

1. Eliminación de la linea base.
2. Ventaneo de los datos.
3. Aplicación del filtro de Butterworth.
4. Detección de los picos.
5. Asignación de instrucciones a los picos detectados.

Para el paso 4 del algoritmo de detección del movimiento horizontal de los ojos, tal y como se muestra en la Figura 8, se hace una resta de las señales en F7 y F8 después de aplicar el filtro de Butterworth, estas señales ya filtradas tienen ya unos cuantos picos, al restar ambas señales obtenemos solo aquellos picos que nos interesan y que determinan la intención del usuario de mover el robot. Los picos que están por debajo de la linea base o cero corresponden a un movimiento ocular hacia la derecha y aquellos que superan el umbral en el sentido positivo corresponden a los ojos moviéndose hacia la izquierda.

Algorithm 1 Algoritmo para la detección del movimiento ocular horizontal Suresh y Schwager, 2016.

1: Restar a los datos la linea base usando una muestra de tamaño $\tau = 640$.
2: Ventanear los datos usando un tamaño de ventana $w \in \mathbb{I}^+$

$$u_i^n(t) = \eta u_i(w) \tag{3}$$

Donde η representa el número de ventana. En nuestro caso $w = 125$ muestras corresponde un segundo de datos sin traslape.
3: Aplicar el filtro de Butterworth de 8° orden pasa bajas de 4 Hz a los datos ventaneados para aislar las señales del movimiento ocular horizontal.
4: Restar las señales resultantes de ambos electrodos.

$$u_{F7-F8}^n(t) = u_{F7}^n(t) - u_{F8}^n(t) \tag{4}$$

5: Detectar los picos y valles usando un umbral de $200\mu V$ y con una separación mínima de $w - 1$ muestras en $u_{F7-F8}^n(t)$.
6: Asignar los picos positivos a el movimiento ocular hacia la izquierda $e_i^n \in \mathbb{Z}^+$ y los valles o picos negativos al movimiento ocular hacia la derecha $e_d^n \in \mathbb{Z}^+$.

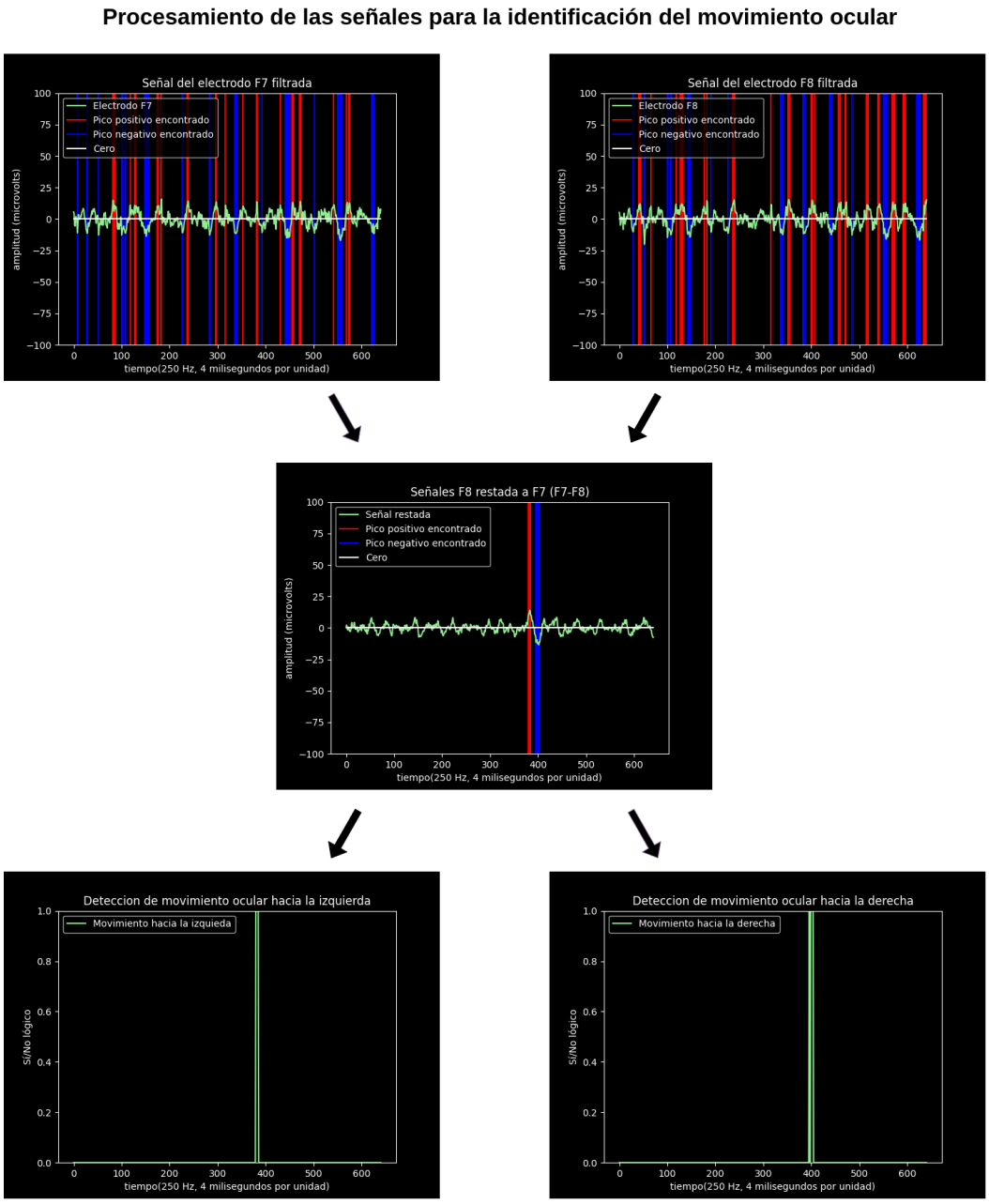

Figura 8: Es posible observar como se detectan picos en ambos electrodos, al restar ambas señales, se reduce el número de picos, estos son detectados por el algoritmo que procesa la señal y la convierte a una instrucción operable por el robot.

3.2 EL ALGORITMO PARA CONTROLAR EL ROBOT

Ya que se usa ROS para controlar el robot, nos atenemos a la estructura de un proyecto en ROS para controlarlo. En ROS se manejan tópicos R. D. Team, 2019a, que son los *buses* a través de los cuales los nodos se mandan mensajes. Para nuestro caso el tópico usado se llama `cmd_vel`, usando este tópico se pueden enviar mensajes al robot del tipo `Twist`, que son mensajes que especifican una velocidad lineal y una angular usando dos vectores de tres dimensiones. El algoritmo que realiza esta comunicación y manda instrucciones al robot es denominado `Publisher` R. D. Team, 2019c o publicador (Algoritmo 2), este algoritmo implementa algunos métodos y bibliotecas de ROS para realizar la conexión y mandar el mensaje. En este caso cuando el Algoritmo 5 detecta un pico manda a llamar al método `talker()` (Algoritmo 2), que es el que funciona como publicador, mandando un mensaje de tipo `Twist` y enviándolo al tópico `cmd_vel` para que sea recogido por el robot.

Los mensajes de este tipo, expresan la velocidad en dos partes, la linear y la angular. Cada una de estas velocidades se expresa en el mensaje como un vector de tres dimensiones (X, Y, Z), para ambos casos, las componentes del vector representan la velocidad en cada uno de los ejes, con la diferencia de que el primer vector representa velocidades lineales y el segundo velocidades angulares, es decir, el primero representa un desplazamiento sobre un plano de tres dimensiones y el segundo una rotación sobre este mismo plano, siendo las unidades metro/segundo y radianes/segundo respectivamente ya que ROS adopta el sistema internacional (SI) R. D. Team, 2019b.

Para nuestra implementación, cuando se detecta un movimiento ocular a la izquierda, se manda a girar el robot hacia la izquierda, si el movimiento se detecta hacia la derecha, la instrucción es que el robot también se mueva hacia la derecha, en cualquiera de ambos casos y también cuando no se detecte ningún movimiento ocular, el robot simplemente sigue avanzando. Para lograr este comportamiento se crearon 3 mensajes, el primero es **Avanzar**, este mensaje mantiene una velocidad lineal de $0{,}3m/s$ sobre el eje X y $0m/s$ en el resto, además, para este mensaje no se manda una velocidad angular. El segundo es el mensaje **GirarIzquierda**, este mantiene el mismo vector de velocidad lineal que el mensaje **Avanzar** y además agrega un vector de velocidad angular de $0{,}3rad/s$ sobre el eje Z y $0rad/s$ sobre el resto para que el robot gire en sentido contrario a las manecillas del reloj. Finalmente el mensaje **GirarDerecha** mantiene los mismos valores del mensaje **GirarIzquierda**, excepto en el eje Z

de su velocidad angular, donde esta es de $-0{,}3 rad/s$ para que el robot gire en el sentido de las manecillas del reloj.

Algorithm 2 Algoritmo para realizar el paso de mensajes al robot virtual usando ROS.

Require: dirección

```
Crear un nodo publicador para el comando cmd_vel
Iniciar el nodo publicador
mensajeAvanzar ← Nuevo objeto Twist
mensajeAvanzar.linear.x = 0.3
mensajeAvanzar.angular.z = 0.0
mensajeGirarIzquierda ← Nuevo objeto Twist
mensajeGirarIzquierda.linear.x = 0.3
mensajeGirarIzquierda.angular.z = 0.3
mensajeGirarDerecha ← Nuevo objeto Twist
mensajeGirarDerecha.linear.x = 0.3
mensajeGirarDerecha.angular.z = -0.3
if dirección = avanzar then
    publicarMensaje(mensajeAvanzar)
else if dirección = izquierda then
    publicarMensaje(mensajeGirarIzquierda)
else if dirección = derecha then
    publicarMensaje(mensajeGirarDerecha)
else
    publicarMensaje(mensajeAvanzar)
end if
```

3.3 RECOLECCIÓN DE LOS DATOS DE EEG USANDO PYOPENBCI

El sistema de recolección *OpenBCI ultracortex biosensing headset* consta de una estructura de soporte que sostienen los electrodos, los electrodos, así como las tarjetas Cyton y Daisy. La tarjeta Cyton que es la principal, además de tener 8 *pines* para lectura de datos de los electrodos, tiene dos salidas a tierra que deben colocarse sobre las orejas del sujeto de pruebas. Los electrodos además, pueden conectarse y desconectarse de las tarjeta Cyton Daisy, también pueden

cambiarse de posición en el casco. Esto se convierte en una ventaja, la de poder usar cualquier electrodo en cualquier posición disponible, al leer los datos desde la computadora, el canal que debamos leer para cierto electrodo dependerá de la posición que se haya elegido al conectar ese mismo electrodo a los *pines* de entrada de las tarjetas Cyton Daisy.

OpenBCI proporciona `pyOpenBCI` una biblioteca para Python que permite recolectar de manera sencilla los datos del casco. Para usarla simplemente se debe especificar el puerto en el que esta conectada a tarjeta USB y si está o no conectado el módulo Daisy. Una vez hecha esta especificación simplemente se tiene que mandar la orden que inicia el flujo de datos (Código 1).

Código 1: La primera instrucción sirve para especificar el puerto que se esta usando así como el uso de la tarjeta Daisy y la segunda hace que inicie el flujo de datos del casco a la computadora.

```
1 board = OpenBCICyton(port='/dev/ttyUSB0', daisy=True)
2 board.start_stream(print_raw_to_file)
```

Las unidades medidas por la tarjeta no vienen dadas en una unidad estandarizada, es necesario convertirlas, en nuestro caso la conversión se realiza a micro volts (μV) esta conversión se realiza multiplicando el valor leído desde el casco por el factor de conversión de la Ecuación 5. Esta factor de conversión viene dado por la propia documentación de OpenBCI para Python O. D. Team, 2021b.

$$\mu VPorUnidad = \frac{\frac{4500000}{24}}{2^23 - 1} \tag{5}$$

Una vez realizada a conversión los datos son escritos en un archivo de datos que es accedido simultáneamente por el algoritmo de detección de movimiento ocular (Algoritmo 1).

3.4 LECTURA DE SEÑALES DESDE ARCHIVO DE DATOS

La función de este modulo es proporcionar al programa las señales provenientes de los electrodos, enviando un arreglo con la señal leída por cada uno de los electrodos cada que el programa principal requiere datos actualizados.

En este caso los datos de EEG se encuentran en un archivo de texto que es escrito por cuando se da la orden de iniciar el flujo de datos en el Código 1.La función de Algoritmo 3 es tomar de ese archivo los renglones que representan los valores leídos en un instante de tiempo y retornarlos en forma de un vector de valores flotantes. En este algoritmo (Algoritmo 3), se representa al archivo de datos como un vector de vectores. Este vector se recorre uno a uno cada que es solicitado y regresa el vector que contiene los valores leídos por el casco en un momento determinado. Cuando se realiza la solicitud de un vector de valores y se ha llegado al final del archivo entonces no hay más vectores para regresar y simplemente se retorna un vector vacío, esto ayuda para que el algoritmo que consume estos datos (Algoritmo 5) no se quede parado esperando a que nuevos datos vengan del casco en caso de alcanzar el la última lectura escrita.

Algorithm 3 Algoritmo para obtener un vector con los valores enviados por el casco.

Require: $archivo, indiceArchivo, limiteIndiceArchivo$

1: **if** $indiceArchivo \leqslant limiteIndiceArchivo$ **then**
2: $\quad vectorValores \leftarrow archivo_{indiceArchivo}$
3: **else**
4: $\quad vectorValores \leftarrow \emptyset$
5: **end if**

3.5 IMPLEMENTACIÓN DE LA ESTRUCTURA TIPO COLA

Poder visualizar los datos en forma gráfica implica tener un rango amplio de datos disponibles en cada momento, la estructura de cola sirve mucho para este propósito ya que los primeros datos en entrar (los más viejos) son los primeros en salir, es decir, los datos guardados en la cola siempre son los más recientes.

Esta estructura contiene las operaciones bien conocidas `push()` y `pop()`, además implementa las funciones `get_as_array()` que devuelve un arreglo con los valores actuales de la cola, esto porque algunas funciones de Python requieren el manejo de los datos en forma de `array` y no como un objeto de la clase `queue` y `clear()` que limpia toda la cola dejándola vacía.

3.6 DETECCIÓN DE LOS PICOS EN LA SEÑAL Y GESTIÓN DEL ROBOT VIRTUAL

En esta sección se describe el módulo principal (Algoritmo 5), el cual integra en su interior al Algoritmo 1 cuya función es detectar los picos que describen el movimiento ocular horizontal. Además de esto, este algoritmo (Algoritmo 5) se encarga de gestionar la comunicación con el módulo `rospy` que es el modulo proporcionado por ROS para comunicarse desde Python con el robot virtual.

Para empezar se da la instrucción al robot de avanzar (aún sin girar). Se crean los *buffers* que contienen los datos leídos desde el archivo al que escribe el casco, un *buffer* para el electrodo F8 y otro para el F7. Estos se llenan cuando llegan 640 muestras, con las cuales podemos calcular el promedio y así sacar la linea base para un segmento de datos (Algoritmo 4). Restamos a los datos en *buffer* el promedio calculado. Procedemos a dividir los datos en ventanas de 125 muestras para aplicar el filtro de Butterworth, de esta manera aplanamos los valles y crestas más largos de la señal, obteniendo una linea más plana que nos permita distinguir los picos producidos por el movimiento ocular horizontal. Realizamos la resta de las señales filtradas de los electrodos F7 y F8. Finalmente procedemos a distinguir los picos y valles que superen los umbrales de $200\mu V$ y $-200\mu V$ respectivamente para asignar los primeros a un movimiento ocular horizontal hacia la izquierda y los segundos hacia la derecha. Una vez determinados los movimientos oculares se llama al algoritmo propio del método `talker()` (Algoritmo 2) que implementa el paso de instrucciones desde el Algoritmo 5 al robot virtual para que el robot cambie su trayectoria.

Algorithm 4 Algoritmo para obtener el promedio de los datos de un vector.

Require: $vector, N$

1: $suma \leftarrow \sum_{i=0}^{N-1} vector_i$

2: $promedio \leftarrow \dfrac{suma}{N}$

Algorithm 5 Algoritmo principal para detectar los picos en la señal y gestionar el robot virtual.

Require: tamañoBuffer
Require: tamañoW
1: talker(avanzar)
2: valoresActuales ← ∅
3: abrirArchivoDeDatos()
4: valoresActuales ← siguienteVector()
5: **while** |valoresActuales| ⩾ 0 **do**
6: i ← 0
7: bufferF7 ← ∅
8: bufferF8 ← ∅
9: **while** |valoresActuales| ⩾ 0 & i ⩽ tamañoBuffer **do**
10: bufferF7.push(valoresActuales$_0$)
11: bufferF8.push(valoresActuales$_1$)
12: i ← i + 1
13: valoresActuales ← siguienteVector()
14: **end while**
15: **if** i ⩽ tamañoBuffer **then**
16: cerrarArchivoDeDatos()
17: terminar()
18: **end if**
19: vectorF7 ← bufferF7
20: vectorF8 ← bufferF8
21: promedioF7 ← promedio(vectorF7)
22: promedioF8 ← promedio(vectorF8)
23: **for** i ← 0 to tamañoBuffer **do**
24: vectorF7$_i$ ← vectorF7$_i$ − promedioF7
25: vectorF8$_i$ ← vectorF8$_i$ − promedioF8
26: **end for**
27: filtradaF7 ← filtroButterWorth(vectorF7)
28: filtradaF8 ← filtroButterWorth(vectorF8)
29: señalRestada ← filtradaF7 − filtradaF8
30: **for** i ← 0 to |señalRestada| **do**
31: **if** señalRestada$_i$ ⩾ 200μV **and** separación mínima al anterior pico es tamañoW **then**
32: w ← 1
33: talker(derecha)
34: **else if** señalRestada$_i$ ⩽ −200μV **and** separación mínima al anterior pico es tamañoW **then**
35: w ← 1
36: talker(izquierda)
37: **end if**
38: **end for**
39: vcectorValores ← siguienteVector()
40: **end while**

4 EXPERIMENTOS

Se realizaron 20 pruebas en para el robot en la simulación, 10 dirigiendo los ojos a la izquierda y 10 a la derecha. cada prueba con una duración de 40 segundos de grabación de EEG. Los resultados se muestran en la Tabla 1. La tabla muestra del lado izquierdo el número de veces que el algoritmo detectó un movimiento ocular a la izquierda y del lado derecho, el número de veces que lo detectó a la derecha. Considéranos que el experimento pasó la prueba si en las sesiones en las que el usuario realizo movimientos oculares a la izquierda el experimento registró un mayor número de movimientos a la izquierda que a la derecha y si en aquellos en los que el usuario realizo movimientos a la derecha el experimento registró un mayor número de movimientos a la derecha que a la izquierda.

las Figuras 9, 10, 11 muestran tres momentos de un experimento aleatorio con el casco. En la Figura 9 se muestra la posición inicial del robot vista desde arriba, la Figura 10 muestra la posición del robot después de algunos movimientos, puede observarse que se ha desplazado a la derecha del punto de origen. Finalmente la Figura 11 muestra la posición en la que termina el robot, ahora se ha desplazado hasta llegar por debajo del punto de origen.

Tabla de experimentos			
Ojos a la izquierda			
Num	Contador izquierda	Contador derecha	Resultado
1	37	35	CORRECTO
2	34	60	FALLO
3	38	28	CORRECTO
4	64	54	CORRECTO
5	42	68	FALLO
6	68	58	CORRECTO
7	47	74	FALLO
8	44	46	FALLO
9	55	66	FALLO
10	44	44	FALLO
Ojos a la derecha			
Num	Contador izquierda	Contador derecha	Resultado
1	70	55	FALLO
2	73	53	FALLO
3	58	46	CORRECTO
4	46	52	CORRECTO
5	61	44	FALLO
6	56	55	FALLO
7	54	53	FALLO
8	32	37	CORRECTO
9	31	44	CORRECTO
10	38	24	FALLO

Tabla 1: Registro de los resultados de los 20 experimentos, 10 realizando movimientos oculares a la izquierda y 10 a la derecha. Para cada prueba se tomaron 40 segundos de grabación de datos de EEG. Se considera que el experimento pasó la prueba si en las sesiones en las que el usuario realizo movimientos oculares a la izquierda el experimento registró un mayor número de movimientos a la izquierda que a la derecha y si en aquellos en los que el usuario realizo movimientos a la derecha el experimento registró un mayor número de movimientos a la derecha que a la izquierda.

Como puede observarse por la Tabla 1, la efectividad del Algoritmo 5 es del 40 % en tanto para detectar movimientos oculares a la izquierda como a la derecha. Esta baja efectividad podría atribuirse a una confusión de los movimientos oculares de retorno, es decir, aquellos en los que el usuario mueve los

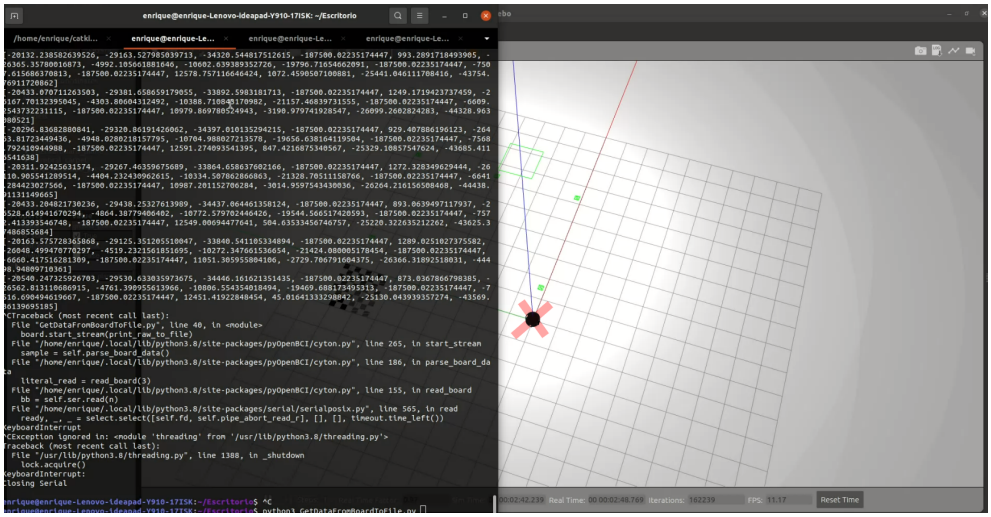

Figura 9: Posición inicial del robot visto desde arriba, se encuentra marcada con una cruz roja el punto de origen.

ojos desde la izquierda o derecha para regresarlos nuevamente a su posición central.

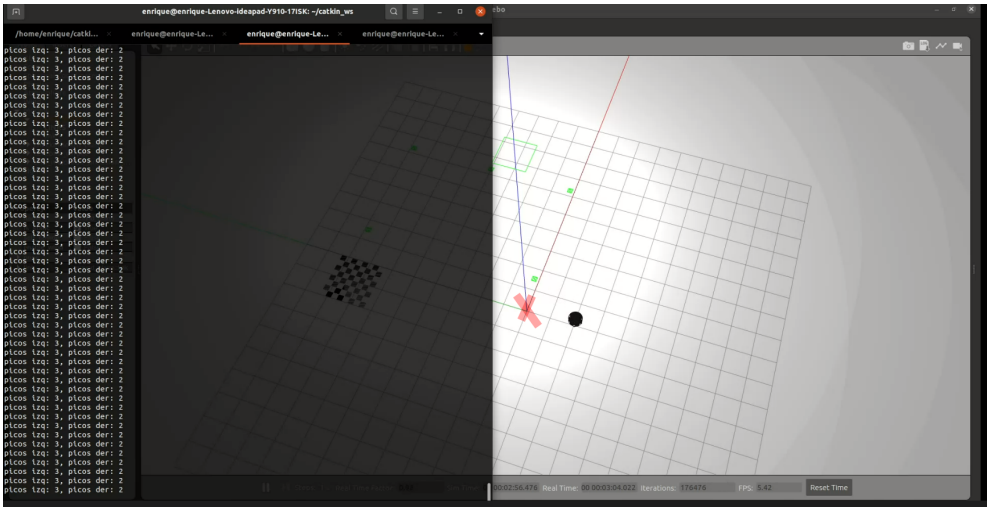

Figura 10: Posición del robot después de algunos movimientos, ahora está a la derecha de la posición de origen marcada con una cruz roja.

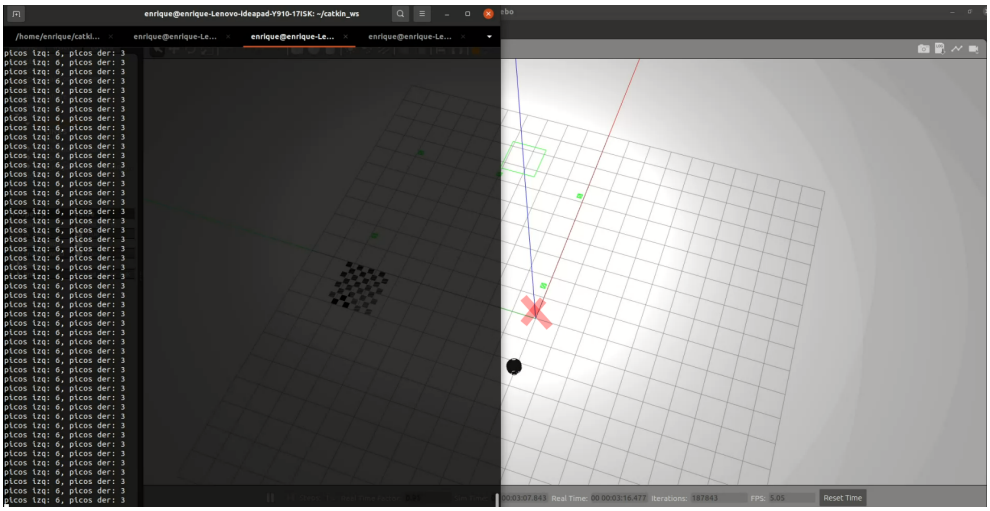

Figura 11: Posición final del robot, se movió nuevamente para quedar situado ahora por debajo de el punto origen marcado con una cruz roja.

5 ANEXOS

5.1 ROS

5.1.1 ¿Qué es ROS?

ROS es un meta-sistema operativo para robots R. D. Team, 2018. Proporciona los servicios que se esperan de un sistema operativo, tales como: abstracción del hardware, control del dispositivo a bajo nivel, implementación de funcionalidades comúnmente utilizadas, paso de mensajes entre procesos, manejo de paquetes. También proporciona herramientas y librerias para obtener, construir, escribir y correr código sobre múltiples computadoras. Es similar a otros Frameworks para robots tales como: Player, YARP, Orocos, CARMEN, MOOS y Microsoft Robotics Studio.

5.1.2 Versionamiento de ROS

ROS, como los sistemas operativos linux, se maneja por distribuciones. El propósito de las distribuciones es mantener un orden relativamente estable sobre el cual los programadores puedan trabajar. Así, una vez que una distribución es liberada, los cambios y ajustes que se hacen en los paquetes son mínimos R. D. Team, 2021a.

Dado que es la principal plataforma en que es usado ROS es Linux, OpenRobotics y Canonical tratan de mantener un mismo paso de liberación de sus versiones. De este modo versión de ROS tiene su contraparte en una versión de Ubuntu. Por ejemplo, se recomienda instalar ROS Melodic Morenia en Ubuntu 18.04 e instalar ROS Noetic Ninjemys en Ubuntu 20.04.

Para este proyecto se usa la tupla ROS Noetic Ninjemys sobre Ubuntu 20.04. siendo ambas las veriones estables más recientes tanto de ROS, como de Ubuntu a la fecha de Agosto de 2021. Teniendo ROS Noetic Ninjemys soporte hasta Mayo de 2025 y Ubuntu 20.04 hasta 2030.

5.1.3 Manual de instalación de ROS Noetic Ninjemys sobre Ubuntu 20.04.

Por R. D. Team, 2021b

1. Configuración de los repositorios Configuración de Ubuntu para que acepte paquetes de software de packages.ros.org.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
    $(lsb_release -sc) main" > /etc/apt/sources.list.d/
    ros-latest.list'
```

2. Configuración de las llaves

```
sudo apt install curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/
    master/ros.asc | sudo apt-key add -
```

3. Instalación

```
sudo apt update
sudo apt install ros-noetic-desktop-full
```

4. Configuración del entorno Se debe correr el siguiente script en cada terminal donde se use ROS

```
source /opt/ros/noetic/setup.bash
```

   O se puede establecer que se corra el script automáticamente cada vez que una nueva terminal es abierta.

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

5. Dependiencias para construir paquetes Para crear y administrar espacios de trabajo propios se pueden instalar los siguientes paquetes.

```
sudo apt install python3-rosdep python3-rosinstall python
    3-rosinstall-generator python3-wstool build-essential
sudo apt install python3-rosdep
sudo rosdep init
rosdep update
```

6 CONCLUSIONES

- La imprecisión del algoritmo para distinguir correctamente los movimientos que corresponden a un movimiento ocular del centro a la izquierda o a la derecha de un movimiento que regresa de derecha o izquierda al centro hacen que el algoritmo no sea apropiado para ocuparse en aplicaciones en tiempo real que requieran de alto grado de precisión o en aquellas en las que el usuario del casco de EEG no tenga que pensar detalladamente sus movimientos musculares al intentar mandar instrucciones.

- El uso de los movimientos oculares como medio para controlar el robot tiene la ventaja de dejar las extremidades del usuario libres, no obstante, tiene el inconveniente de que al mover los ojos el usuario aparta la mirada de su objeto de atención, esto provoca que pueda cometer algunos errores al distraerse de aquello que esta controlando.

- Los movimientos oculares no son los únicos que se ven reflejados en la señal de EEG, las contracciones fuertes de los músculos de la cara como las producidas al apretar la mandíbula también son visibles como picos en la señal, esto puede considerarse una desventaja si no se desea tomar en cuenta estas señales ya que generan ruido, por el contrario si se toman en cuenta pueden representar una ventaja al brindarnos más formas de controlar las instrucciones que se mandan al robot, esto gracias a que los picos producidos por estas son mucho más grandes y perfectamente distinguibles de aquellos producidos por el movimiento ocular.

BIBLIOGRAFÍA

Ballou, Glen

2013 *Handbook for sound engineers*, Taylor & Francis.

Barreto Galeano, Johanna Paola

2014 "Electroencefalograma", *Comportamiento de Elección y Procesos de Decisión*.

Bianchi, G. y R. Sorrentino

2007 *Electronic Filter Simulation & Design*, McGraw-Hill Education, ISBN: 9780071494670, https://books.google.com.mx/books?id=5S3LCIxnYCcC.

Book: Celestial Mechanics (Tatum)

2021 [Online; accessed 2022-01-03].

Brown, Malcolm, Michael Marmor, Vaegan, Eberhard Zrenner, Mitchell Brigell y Michael Bach

2006 "ISCEV Standard for Clinical Electro-oculography (EOG) 2006", *Documenta Ophthalmologica*, 113, 3 (nov. de 2006), págs. 205-212, ISSN: 1573-2622, DOI: 10.1007/s10633-006-9030-0, https://doi.org/10.1007/s10633-006-9030-0.

Ellis, George

2012 "Chapter 9 - Filters in Control Systems", en *Control System Design Guide (Fourth Edition)*, ed. por George Ellis, Fourth Edition, Butterworth-Heinemann, Boston, págs. 165-183, ISBN: 978-0-12-385920-4, DOI: https://doi.org/10.1016/B978-0-12-385920-4.00009-6, https://www.sciencedirect.com/science/article/pii/B9780123859204000096.

Faraday, Michael

1834 "On Electrical Decomposition", *Philosophical Transactions of the Royal Society*, 124, págs. 77-122, DOI: 10.1098/rstl.1834.0008.

Girón, M.R.O. y Universidad de Córdoba. Departamento de Física Aplicada

1992 *Lecciones de física*, Lecciones de Física, v. 4, Manuel R. Ortega Girón, ISBN: 9788460444459, https://books.google.com.mx/books?id=SBRdAAAACAAJ.

González, Alicia Montserrat Alvarado
2016 Tesis doct., Universidad Autónoma de México, Ciudad de México, México.

Hamming, Richard Wesley
1998 *Digital filters*, Courier Corporation.

Kalagi, Sunil, José Machado, Vitor Carvalho, Filomena Soares y Demétrio Matos
2017 "Brain computer interface systems using non-invasive electroencephalogram signal: A literature review", en *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, págs. 1578-1583, DOI: 10.1109/ICE.2017.8280071.

Kıymık, M.Kemal, İnan Güler, Alper Dizibüyük y Mehmet Akın
2005 "Comparison of STFT and wavelet transform methods in determining epileptic seizure activity in EEG signals for real-time application", *Computers in Biology and Medicine*, 35, 7, págs. 603-616, ISSN: 0010-4825, DOI: https://doi.org/10.1016/j.compbiomed.2004.05.001, https://www.sciencedirect.com/science/article/pii/S0010482504000691.

Kübler, Andrea
2000 "Brain-computer interface technology: A review of the second international meeting", *IEE Transactions on Neural Systems and Rehabilitation Engineering*.
2020 "The history of BCI: From a vision for the future to real support for personhood in people with locked-in syndrome", *Neuroethics*, 13, 2, págs. 163-180.

Morley, Andrew, L Hill y AG Kaditis
2016 "10-20 system EEG Placement", *European Respiratory Society, European Respiratory Society*.

Munyon, Charles N.
2018 "Neuroethics of Non-primary Brain Computer Interface: Focus on Potential Military Applications", *Frontiers in Neuroscience*, 12, pág. 696, ISSN: 1662-453X, DOI: 10.3389/fnins.2018.00696, https://www.frontiersin.org/article/10.3389/fnins.2018.00696.

Novo-Olivas, Carlos, Leticia Guitiérrez y José Bribiesca
2010 "Mapeo Electroencefalográfico y Neurofeedback", en, págs. 371-412, ISBN: 978-970-764-911-8.

Orient-López, F.

2006 "Tratamiento neurorrehabilitador de la esclerosis lateral amiotrófica", *Revista de Neurología*.

Pfurtscheller, Gert, Brendan Allison, Günther Bauernfeind, Clemens Brunner, Teodoro Solis Escalante, Reinhold Scherer, Thorsten Zander, Gernot Mueller-Putz, Christa Neuper y Niels Birbaumer

2010 "The hybrid BCI", *Frontiers in Neuroscience*, 4, pág. 3, ISSN: 1662-453X, DOI: 10.3389/fnpro.2010.00003, https://www.frontiersin.org/article/10.3389/fnpro.2010.00003.

Schalk, Gerwin, Jan Kubanek, Kai Joshua Miller, Nicholas R. Anderson, Eric C. Leuthardt, Jeffrey G. Ojemann, David D. Limbrick, Daniel W. Moran, Lester A. Gerhardt y Jonathan R. Wolpaw

2007 "Decoding two-dimensional movement trajectories using electrocorticographic signals in humans." *Journal of neural engineering*, 4 3, págs. 264-75.

Schalk, Gerwin, Dennis J. McFarland, Thilo Hinterberger, Niels Birbaumer y J. R. Wolpaw

2004 "BCI2000: a general-purpose brain-computer interface (BCI) system", *IEEE Transactions on Biomedical Engineering*, 51, págs. 1034-1043.

Smith, J.O.

2007 *Introduction to Digital Filters: With Audio Applications*, Music signal processing series, W3K, ISBN: 9780974560717, https://books.google.com.mx/books?id=pC1iCQUAsHEC.

Suresh y Schwager

2016 "Brain-Swarm Interface (BSI): Controlling a Swarm of Robots with Brain and Eye Signals from an EEG Headset", *arXiv preprintarXiv:1612.08126v1*.

Team, OpenBCI Documentation

2021a *Cython Specs*, OpenBCI, https://docs.openbci.com/Cyton/CytonSpecs.

2021b *SoftwareDevelopment*, OpenBCI, https://docs.openbci.com/ForDevelopers/SoftwareDevelopment/.

Team, ROS Documentation

2018 *Introduction*, ROS, http://wiki.ros.org/ROS/Introduction.

Team, ROS Documentation

2019a *Topics*, ROS, http://wiki.ros.org/Topics.

2019b *Twist Message*, ROS, http://docs.ros.org/en/lunar/api/geometry_msgs/html/msg/Twist.html.

2019c *Writing a Simple Publisher and Subscriber (Python)*, ROS, http://wiki.ros.org/Topics.

2021a *Distributions*, ROS, http://wiki.ros.org/Distributions.

2021b *Installation*, ROS, http://wiki.ros.org/ROS/Installation.