UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de ingeniería

INTERFACES BIOLÓGICAS HUMANO-MÁQUINA

Trabajo de graduación en modalidad de megaproyecto presentado por

Oscar Fernando Castañeda Fernández

para optar al grado académico de Licenciado en Ingeniería Electrónica y

Javier Iván Castillo Rivera

Pedro Iván Castillo Rivera

Diego Mario Alejandro García Maldonado

para optar al grado académico de Licenciados en Ingeniería Mecatrónica

Guatemala,

2015

INTERFACES BIOLÓGICAS HUMANO-MÁQUINA

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de ingeniería

INTERFACES BIOLÓGICAS HUMANO-MÁQUINA

Trabajo de graduación en modalidad de megaproyecto presentado por

Oscar Fernando Castañeda Fernández

para optar al grado académico de Licenciado en Ingeniería Electrónica y

Javier Iván Castillo Rivera

Pedro Iván Castillo Rivera

Diego Mario Alejandro García Maldonado

para optar al grado académico de Licenciados en Ingeniería Mecatrónica

Guatemala,

2015

Vo. Bo. :

(f) _____

MSc. Roberto Enrique Saravia Fernández
Coordinador

Directores de los estudiantes que trabajaron el Megaproyecto:

(f) _____

MSc. Carlos Alberto Esquit Hernández
Director de Ingeniería Electrónica y Mecatrónica

Fecha de aprobación: Guatemala, 25 de noviembre del 2015.

PREFACIO

El megaproyecto interfaces biológicas humano-máquina busca dar continuidad a una línea de investigación que ha sido previamente estudiada dentro de la Universidad del Valle de Guatemala, la cual es el desarrollo de interfaces no convencionales, basadas en señales generadas por el cuerpo humano, para permitir al usuario el envío de comandos a algún dispositivo electrónico. Este es un campo de sumo interés a nivel mundial y del cual podrían surgir las interfaces humano-máquina del futuro, que permitirán una comunicación más natural y eficiente entre el ser humano y los equipos que lo rodean. Por tanto, es de suma importancia continuar el desarrollo de esta investigación dentro de la Universidad del Valle de Guatemala, para estar al tanto de las tendencias en el área y, así, proponer nuevas alternativas.

Este trabajo llevó a cabo la implementación de tres sistemas: El primero de ellos fue una interfaz cerebro-computadora, basada en el potencial relacionado a eventos conocido como onda P300, a partir del cual se puede identificar el carácter en el cual el usuario está concentrado, permitiendo escribir únicamente con la mente. El segundo sistema consistía en el reconocimiento de cuatro posiciones de cada una de las manos a partir de las señales electromiográficas, las cuales fueron utilizadas para controlar un cuadcóptero. Finalmente, el tercer sistema correspondía a la detección de cinemática de extremidades por medio de procesamiento de imágenes. A partir de dicha información, los movimientos realizados por el usuario eran replicados por el robot humanoide NAO.

En todos estos sistemas, fuimos capaces de aplicar los conocimientos adquiridos a lo largo de nuestra carrera, así como también fuimos capaces de aprender nuevos conocimientos para aplicarlos en la realización de este trabajo. Consideramos que los resultados fueron satisfactorios y esperamos que sean de utilidad para futuras investigaciones.

Deseamos agradecer a todas las personas que nos apoyaron durante la realización de este trabajo y que contribuyeron, de forma directa o indirecta, con su tiempo, conocimientos y guía. Deseamos agradecer especialmente a nuestras familias, por su apoyo a lo largo de todo este proceso, así como a nuestro asesor MSc. Roberto Saravia, por su constante guía y sugerencias.

ÍNDICE

LISTA DE CUADROS

Página

LISTA DE FIGURAS

Página

RESUMEN

El megaproyecto “Interfaces biológicas humano-máquina” tiene como objetivo desarrollar diversos tipos de interfaces, las cuales utilizan señales provenientes del cuerpo para lograr comunicación entre humano y máquina. Estos se trabajaron a partir de señales de EEG (electroencefalografía), EMG (electromiografía), y la cinemática de las extremidades superiores por medio del procesamiento de imágenes. El proyecto se dividió en tres sistemas donde se implementaban cada una de las señales mencionadas anteriormente.

Las señales de EEG, en conjunto con el teclado P300, permiten que la mayoría de personas escriban con la mente. El teclado P300 consiste en una matriz que contiene caracteres alfanuméricos, los cuales se iluminan aleatoriamente. El usuario debe fijar su mirada en el carácter que desea escribir y contar cada vez que este se ilumina. Tal acción genera una reacción en el cerebro, conocida como onda P300, la cual puede ser capturada utilizando un equipo de adquisición de EEG. Aplicando algoritmos de clasificación, puede determinarse si, luego de ocurrida una iluminación en la pantalla, el usuario presentó o no la onda, información con la cual se puede deducir el carácter que desea escribir y mostrarlo en el monitor.

Para implementar un teclado P300 se requieren de cuatro tareas: la estimulación del usuario, la grabación de sus señales de EEG, el procesamiento de estas y su clasificación. Por tanto, en este trabajo se contemplaron los aspectos relacionados a la obtención de señales electroencefalográficas y al desarrollo de una interfaz cerebro computadora que mediante estímulos visuales en forma de luz, permitía implementar dichas señales para escribir una palabra en pantalla. También se exploraron distintas cadenas de procesamiento, conformadas por filtros temporales y espaciales, etapas de normalización, decimación y promedio de señales. El objetivo de dichas cadenas era facilitar la tarea de clasificación a un algoritmo de aprendizaje automático. Se utilizó un ensamble de tres máquinas de vectores de soporte (SVM) de *kernel* gaussiano como clasificador, debido a su capacidad de representar fronteras no lineales y de operar sobre datos de gran dimensión. Como resultado, se obtuvo un sistema que podía predecir caracteres con una exactitud promedio 1.5 veces mayor que la dada por un programa de distribución libre.

Por otra parte, las señales EMG de los músculos flexores y extensores del antebrazo fueron utilizadas para clasificar los movimientos o intentos de movimientos de las manos. Dichas señales fueron obtenidas utilizando cuatro electrodos para medir las señales EMG, con las cuales se buscaban clasificar cuatro movimientos diferentes. La clasificación fue llevada a cabo utilizando algoritmos de inteligencia artificial,

dentro de los cuales se realizaron pruebas utilizando SVM, regresión logística y red neuronal. SVM obtuvo un rendimiento máximo de 96% y regresión logística obtuvo un máximo de 77%. A pesar del buen rendimiento de SVM, se utilizó una red neural pues tenía un rendimiento arriba de 90% y era más estable que SVM. Para dar una aplicación a dichas señales, se controló un cuadcóptero (AR Drone 2.0).

Finalmente, el sistema de detección de cinemática de las extremidades superiores por medio del procesamiento de imágenes consistió en utilizar las coordenadas cartesianas de las extremidades del humano, obtenidas con procesamiento de imágenes, para realizar la cinemática del robot humanoide NAO. El procesamiento de imágenes realizado era capaz de operar sin importar el entorno en el cual es utilizado. El robot humanoide NAO era capaz de replicar los movimientos del usuario con un error máximo de 5 cm en uno de los ejes y 2 cm en los otros dos ejes. Únicamente se modelaron los brazos de NAO con los parámetros Denavit-Hartenberg. A pesar de que se controlaron los ángulos de la cabeza, esto se hizo sin implementar la cinemática inversa, sino basándose únicamente en los ángulo proveídos por el sistema basado en visión por computadora, por lo que únicamente se implementó la cinemática directa en las articulaciones de la cabeza.

I. INTRODUCCIÓN

Las interfaces biológicas humano-máquina son un área de investigación que ha adquirido un fuerte auge en los últimos años, debido a las aplicaciones que puede tener para mejorar la calidad de vida de personas que sufren de alguna discapacidad motora, así como las posibilidades que ofrecen para hacer de la interacción con distintos dispositivos una más intuitiva y eficiente. Este megaproyecto trabajó en la implementación de diferentes interfaces biológicas humano-máquina: interfaz cerebro computadora, interfaz cuerpo computadora y visión por computadora.

Las interfaces cerebro computadora permiten el control de dispositivos electrónicos a través de las señales eléctricas generadas por la actividad neuronal del cerebro humano. Una de las interfaces cerebro computadora más estudiadas es el teclado P300, el cual se aprovecha de una onda generada por la aparición de un estímulo objetivo de baja probabilidad de ocurrencia dentro de un conjunto de estímulos. Dicha onda puede ser utilizada para determinar el momento en que una persona reaccionó frente al estímulo deseado, lo cual se puede mapear, por ejemplo, a una letra a escribir en la computadora.

A pesar de que el teclado P300 ha sido estudiado alrededor del mundo desde 1988, en nuestro país esta aplicación, así como las interfaces cerebro computadora en general, es un tema en el cual se cuenta con poco conocimiento y experiencia. Sin embargo, el teclado P300 es una de las interfaces cerebro computadora más analizadas, por lo que constituye un buen punto de inicio para impulsar un campo de investigación en esta área. Es por ello que, dentro del megaproyecto "Interfaces biológicas humano-máquina", se planteó el desarrollo de una interfaz cerebro computadora cuyo objetivo era implementar un teclado P300.

Para ello, dicha interfaz cerebro computadora fue dividida en dos módulos: Uno de ellos para estimular al usuario y grabar sus señales de electroencefalografía y otra, para analizar y clasificar dichas señales. Como resultado, se pudo implementar un teclado P300, con algoritmos de procesamiento y clasificación tanto para el análisis de las señales fuera de línea, como en tiempo real. Si bien la exactitud del sistema es difícil de evaluar puesto que depende fuertemente del usuario y el entorno en el cual las grabaciones son realizadas, se logró implementar un sistema que tenía una exactitud promedio 1.5 veces mayor que la de un clasificador disponible como un programa de distribución libre. Sin embargo, hace falta más trabajo para poder validar este sistema con personas que sufren de discapacidades motoras, para que sea lo suficientemente robusto para su uso en la vida cotidiana, razón por la cual se considera a este trabajo como un primer paso en esta línea de investigación.

Por otra parte, la interfaz cuerpo computadora desarrollada en este trabajo se basa en señales de electromiografía (EMG), las cuales son señales eléctricas que se generan mediante la actividad muscular.

Esta puede ser superficial o intramuscular, dependiendo si los electrodos son no invasivos o invasivos, respectivamente. Por ello nos limitamos a trabajar con la EMG superficial [76]. Se utilizaron modelos de inteligencia artificial para clasificar las señales EMG captadas y posteriormente controlar un cuadcóptero (AR Drone 2.0). Se realizaron pruebas con tres modelos diferentes: Redes neurales artificiales (ANN), Máquina de Vectores de Soporte (sus siglas en inglés: SVM) y regresión logística. Estos modelos son métodos utilizado para aprender patrones y brindar una o varias respuestas en base a ellos [78].

Se compararon los resultados obtenidos con los distintos modelos de inteligencia artificial. En cada uno de ellos se analizaron parámetros de muestreo y entrenamiento. Además se analizó el dominio del tiempo y la frecuencia, para determinar qué modelo es el más óptimo y permite reconocer una mayor cantidad de comandos. Se comparó el rendimiento máximo obtenido con cada modelo: SVM obtuvo un máximo de 96%; regresión logística un máximo de 77% y las redes neurales una exactitud arriba de 90%. Se concluyó que las redes neurales presentaban una mayor estabilidad en sus resultados por lo que se utilizó este modelo.

Para esta interfaz se midieron señales EMG en el antebrazo, variando los gestos de la mano. En total se lograron clasificar cuatro diferentes posiciones en cada mano. Cada posición realizaba un comando distinto en el cuadcóptero.

El sistema de visión por computadora fue la tercera interfaz humano-máquina desarrollada. La visión por computadora es la percepción en tres dimensiones que se obtiene al momento de analizar la forma y apariencia de los objetos, por medio de imágenes. Con esta se puede llegar a rastrear a una persona en ambientes complejos, como detectar rostros, bordes y colores, entre otras cosas [86]. Obteniendo las posiciones del cuerpo humano se pueden replicar en robots humanoides como NAO: este es un robot de 58cm que permite crear múltiples aplicaciones con movimiento, audio y reconocimiento de imágenes [4]. Para realizar una interfaz universal el procesamiento se realiza sobre una computadora y se transmite la información de manera inalámbrica.

Para el control de NAO, se modeló al mismo como una cadena, es decir, como un conjunto de eslabones y articulaciones. Se utilizó la notación Denavit-Hartenberg para simplificar la representación de este modelo. Posteriormente se calculó la cinemática inversa de NAO utilizando un método numérico [17]. Se calibraron las constantes de este comparando la medición obtenida del sistema de visión por computadora y los resultados del método numérico. Para evitar que NAO chocara, se simularon los resultados de primero. Luego de corregir el método, se dejó de utilizar el simulador y se implementó el robot físico. Se logró calcular la cinemática inversa de las posiciones obtenidas en el sistema basado en visión por computadora y replicar estos movimientos en NAO con un error máximo de 5 cm.

II. OBJETIVOS

A. OBJETIVO GENERAL

- Desarrollar interfaces no convencionales basadas en señales provenientes del cuerpo para lograr comunicación entre humano y máquina.

B. OBJETIVOS ESPECÍFICOS

- Implementar una cadena de procesamiento y clasificación que permita optimizar el funcionamiento de una interfaz cerebro máquina basada en un sistema P300.
- Obtener, preparar e integrar señales electroencefalográficas con un sistema de inteligencia artificial para desarrollar una aplicación que permita a personas con discapacidades comunicarse por un medio escrito sin necesidad de involucrar actividad motora.
- Evaluar métodos de inteligencia artificial para clasificar señales EMG y utilizar estas señales y las obtenidas del sistema basado en visión por computadora para controlar un robot humanoide y un drone.
- Evaluar métodos de inteligencia artificial para clasificar señales EMG y desarrollar un sistema basado en visión por computadora para proveer información del movimiento del usuario para controlar un dron y un robot humanoide.

III. JUSTIFICACIÓN

En la actualidad existen diversos equipos y máquinas para facilitar actividades a los humanos, cuyas interfaces dependen de ciertas habilidades motrices. Sin embargo, existen personas con padecimientos físicos los cuales les impiden poder utilizar las interfaces ya existentes, por ello se desarrollaron diversos módulos universales, con el objetivo de facilitar dicha interfaz tanto a personas con padecimientos físicos y sin ellos.

En anteriores ocasiones, se han desarrollado, dentro de la Universidad del Valle de Guatemala, interfaces cerebro computadora, pero estas nunca han sido utilizadas fuera del ámbito académico, a pesar de los beneficios que pueden brindar a muchas personas en nuestra sociedad. Esto ha ocurrido por razones de confiabilidad y facilidad de reproducción del sistema. Por tanto, se propone realizar una implementación del teclado P300, una de las interfaces cerebro computadora más estudiadas en el mundo, la cual, con el desarrollo adecuado, podría reproducirse y ser utilizada por pacientes reales de nuestro país.

El módulo de procesamiento y clasificación de señales se encarga, como su nombre lo indica, de dos tareas sumamente importantes: La primera de ellas es el procesamiento de las señales, cuyo propósito es el de mejorar la relación de señal a ruido de los datos capturados por el módulo de obtención de señales, para facilitar la segunda tarea: la clasificación. Esta segunda tarea consiste en poder discriminar, dentro de las señales capturadas, cuál corresponde a una onda P300 o no, para así determinar el carácter en el cual el usuario estaba concentrado.

Las señales de electroencefalografía (EEG) se caracterizan por tener una relación de señal a ruido pequeña, debido a que se generan en el cerebro y deben de atravesar el cráneo para poder ser detectadas por electrodos colocados en el cuero cabelludo. Dicha relación de señal a ruido se ve perjudicada si la señal de interés no son todas las ondas cerebrales, sino solo ciertas componentes de estas, debido a que dentro del cerebro existen millones de neuronas, cada una con su propia actividad eléctrica que se superpone a la de las demás, dificultando la tarea de aislar las señales provenientes de una región específica. La situación es todavía peor si se considera que los movimientos musculares de una persona, tales como el parpadeo o el movimiento de la cabeza y mandíbula, también producen señales eléctricas que son captadas en la electroencefalografía. Todas las características anteriores hacen necesario el procesamiento de las señales para mejorar la relación de señal a ruido y así permitir un adecuado análisis de estas.

Por otra parte, las características de las señales cerebrales varían de sujeto en sujeto en parámetros tales como la latencia y la amplitud, y estas inclusive pueden variar para un mismo sujeto en diferentes sesiones de electroencefalografía. Esto quiere decir que hallar características de las señales de electroencefalografía que permitan clasificarlas para muchos usuarios es una tarea compleja, por lo que se deben de crear modelos

de clasificación propios para cada usuario. Lo anterior hace recomendable el uso de algoritmos clasificadores que puedan, a partir de un conjunto de señales pertenecientes a un usuario, determinar las características que diferencian a un tipo de señal del otro, para así crear un modelo de clasificación. La etapa de clasificación es, entonces, necesaria, ya que sin ella no se podría distinguir las señales que contienen a una P300 de las que no, haciendo imposible la determinación del carácter deseado por el usuario.

El módulo de obtención de señales EEG también es necesario. Como se explicó anteriormente, existe ruido que se genera en las señales EEG, por lo tanto es necesario utilizar herramientas que permiten la captación de estas señales sin inducir más ruido. Por ello es necesario probar diferentes equipos de captación de señales EEG y determinar cuál es el óptimo. Estas señales posteriormente se deben poder implementar en el programa de aplicación de teclado P300, que se desarrolló en MATLAB, para deletrear las letras que el usuario desee. Se desea que el desempeño obtenido con este programa de teclado P300 sea mejor al desempeño de otras aplicaciones existentes.

Para implementar un módulo de captación e interpretación de señales EMG, es necesario implementar un modelo de inteligencia artificial, para clasificar e identificar los diferentes patrones en las señales. Esto se debe a que en el antebrazo existen regiones donde se concentran distintos músculos, lo que introduce ruido en las mediciones y cruce de señales. Una vez identificadas las señales se les puede asignar comandos como subir, bajar, rotar, avanzar, entre otros, e implementarlos en diferentes dispositivos como una computadora, un brazo robótico, un drone, etc.

Para poder controlar robots basados en la simetría humana es necesario implementar un segundo sistema, el cual permita identificar las posiciones en un plano cartesiano de cada articulación: cabeza, hombros, codos, manos, etc. También es necesario para normalizar y escalar las posiciones de las mismas y así poder trabajar con robots imitadores con distintas proporciones a las del cuerpo humano, a través de un sistema de visión por computadora que permita extraer esta información. Esta interfaz permitirá controlar robots de búsqueda, realizar operaciones complejas a distancia o interactuar con un entorno automatizado a través de movimientos gruesos.

Posteriormente, para poder imitar los movimientos, es necesario realizar un análisis cinemático de las coordenadas cartesianas proveídas por el procesamiento de imágenes para convertir estas coordenadas en movimientos de un robot humanoide. Para esto es necesario calcular la cinemática inversa para poder accionar los motores del robot humanoide de tal modo que sus extremidades alcancen las posiciones deseadas. Esto crea un control más intuitivo del robot humanoide NAO.

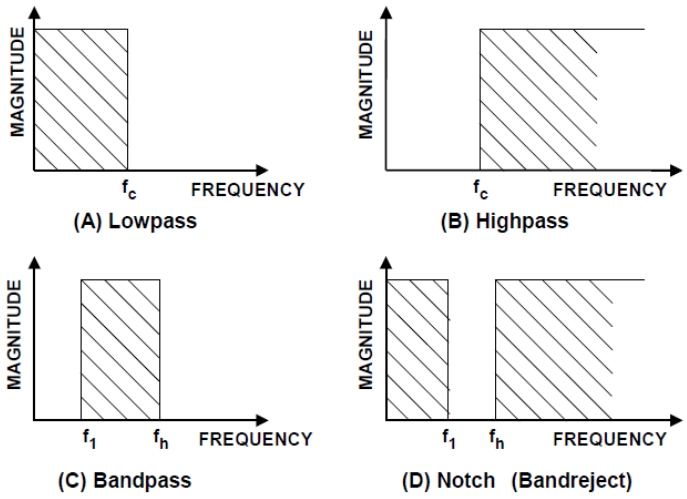
IV. MARCO TEÓRICO

A.FILTROS ANALÓGICOS

Filtros son redes que procesan señales según su frecuencia. Los filtros pueden ser usados para separar las señales, permitiendo pasar las frecuencias de interés y atenuando las demás. Una aplicación donde se utilizan filtros es en un radio, donde solamente se permite el paso de la frecuencia de interés y se eliminan las demás [5]. Sin el uso de filtros se escucharían todas las estaciones al mismo tiempo.

Un filtro ideal posee una ganancia de 1 en las frecuencias de interés o banda de paso, y posee una ganancia de 0 en todas las demás frecuencias, es decir, en la banda de rechazo. La frecuencia donde ocurre el cambio de banda de paso a banda de rechazo se conoce como frecuencia de corte [5]. En la Figura 1 se muestran las cuatro respuestas generales de filtros ideales que se pueden presentar.

Figura 1. Respuestas de filtros ideales [5].



Existe el filtro pasa-baja, el cual permite el paso de las frecuencias debajo de la frecuencia de corte y elimina las demás. El filtro pasa-alta es el complemento del filtro pasa-baja, ya que permite el paso de las frecuencias mayores a la frecuencia de corte y elimina las demás. Si los dos filtros anteriores se conectan en cascada, se consigue un filtro pasa-banda, el cual permite el paso de un rango de frecuencias que se encuentran arriba de la frecuencia de corte del pasa-alta pero debajo de la frecuencia de corte del pasa-baja. El complemento del pasa-banda es el rechaza-banda, el cual atenúa las frecuencias en el rango mencionado anteriormente [5].

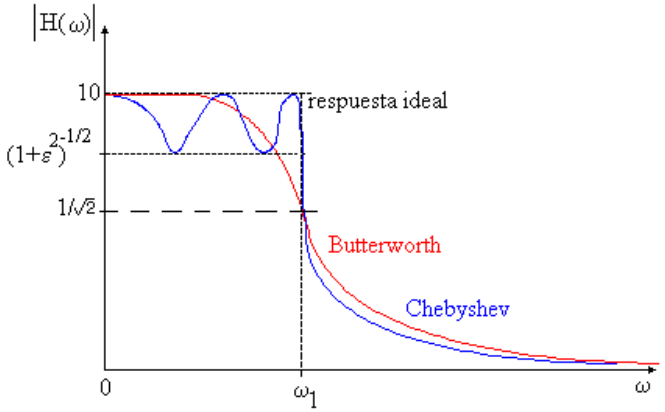
En la práctica, la transición entre la banda de paso y la de rechazo no ocurre instantáneamente, como se muestra en la Figura 1, sino que existe una región de transición. Tampoco poseen una amplitud exactamente de unidad en la banda de paso; algunos varían su magnitud en la banda de paso. Existen diferentes tipos de filtros que se pueden construir, cada uno posee respuestas distintas, las cuales pueden beneficiar o no al sistema. Algunos de los filtros más utilizados son los filtros Butterworth y los filtros Chebyshev [5].

1. Filtro Butterworth. Las características principales de un filtro Butterworth es que posee una ganancia constante en su banda de paso. Esto permite filtrar la señal sin distorsionarla. Una desventaja de este filtro es que posee una región de transición más amplia que otros filtros. Por lo tanto, el cambio entre la banda de paso a la banda de rechazo ocurre paulatinamente, lo que implica que no se eliminan todas las frecuencias en la banda de rechazo [5].

Si la aplicación requiere que no se modifique la ganancia de la señal de entrada, se utilizaría un filtro Butterworth, aunque no se eliminen todas las frecuencias en la banda de rechazo. Se puede modificar el orden del filtro para reducir la región de transición, sin embargo, el circuito a construir sería más complejo y costoso [5].

2. Filtro Chebyshev. Este tipo de filtro posee una región de transición más angosta que el filtro Butterworth. Por lo tanto, elimina más frecuencias de la banda de rechazo que un filtro Butterworth. Sin embargo, su ganancia en la banda de paso no se mantiene constante para todas las frecuencias [5]. Esto altera la amplitud de la señal de entrada, por lo que no se recomienda este filtro si la amplitud de la señal en la banda de paso es crítica. Si la aplicación requiere de una región de transición más angosta, sin importar que se modifique la amplitud de la señal de entrada, entonces sí se recomienda utilizar este filtro.

Figura 2. Comparación filtro Butterworth y Chebyshev [95].

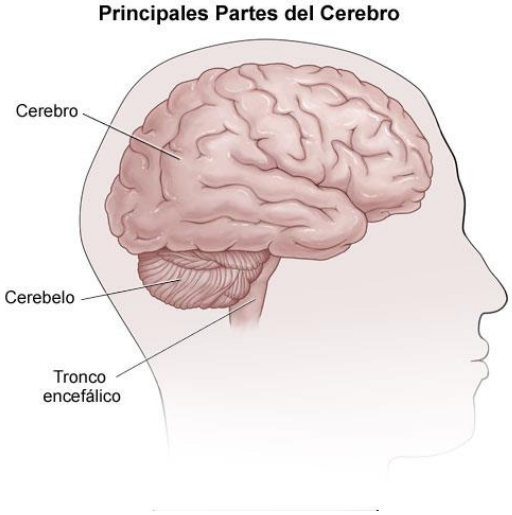


La figura anterior muestra la respuesta de los dos tipos de filtros mencionados. También se logra observar la respuesta ideal del filtro, por lo que se puede apreciar las diferencias entre cada tipo de filtro.

B. EL ENCÉFALO

El encéfalo es la parte más voluminosa del sistema nervioso central (SNC). Es un órgano muy importante, ya que controla todos los procesos que regulan el cuerpo. Algunos de los aspectos que controla son: la memoria, las emociones, el tacto, la actividad motora, la vista y la respiración, entre otros. El encéfalo se puede dividir en tres partes principales: el cerebro, el cerebelo y el tronco del encéfalo [91].

Figura 3**.** Partes principales del encéfalo [91]**.**



1. El tronco del encéfalo o tallo cerebral. Las funciones del tronco del encéfalo incluyen: el movimiento de los ojos y de la boca, la transmisión de mensajes sensoriales (calor, dolor, ruido, etc.), el hambre, la respiración, la función cardíaca, los movimientos musculares involuntarios, los estornudos, la tos, los vómitos y la deglución [91].

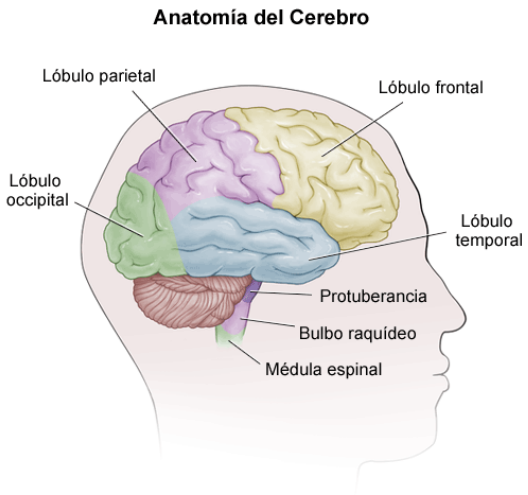
2. El cerebelo. Está ubicado en la parte posterior de la cabeza. Entre sus funciones principales se encuentran: coordinar los movimientos musculares voluntarios y mantener la postura, la estabilidad y el equilibrio [91].

3. El cerebro. El término cerebro se suele utilizar incorrectamente para referirse a la totalidad del contenido del cráneo, el cual en realidad se llama encéfalo. El cerebro es la parte más grande del encéfalo y se divide en dos hemisferios, el derecho y el izquierdo. Algunas investigaciones han demostrado que el hemisferio izquierdo del cerebro está asociada con habilidades lógicas y el hemisferio derecho con la creatividad [82]. Las funciones del cerebro incluyen: iniciación y coordinación de movimientos,

implementación de los cinco sentidos (tacto, vista, oído, etc.), el razonamiento, la resolución de problemas, las emociones y el aprendizaje [91].

a. Áreas de concentración del cerebro. La corteza cerebral es una capa fina de neuronas situadas en la periferia del cerebro, el cual contiene muchas fisuras que aumentan su área superficial. Posiblemente la corteza cerebral sea la subdivisión más importante del encéfalo, ya que contiene 9 de los 12 billones de neuronas en el cerebro humano. Las fisuras más profundas se denominan surcos y algunos se utilizan como límites para dividir la corteza en ciertos lóbulos [6]. El cerebro se divide en cuatro lóbulos, las cuales son: el lóbulo frontal, el lóbulo parietal, el lóbulo occipital y el lóbulo temporal [82]. En la siguiente figura se observa dónde se encuentra cada uno de estos lóbulos.

Figura 4. Lóbulos cerebrales [91].



1) Lóbulo frontal. Es la porción más voluminosa del encéfalo y se encuentra situado en la parte delantera de la cabeza [91]. Está asociado con el razonamiento, el habla, la planeación, los movimientos de los ojos, las emociones y la resolución de problemas [82].

Los lóbulos frontales se consideran como el centro de control de la personalidad. El lóbulo frontal izquierdo está involucrado en capacidades verbales, mientras que el lóbulo frontal derecho está involucrado en capacidades no verbales. Las personas con daño en estos lóbulos exhiben poca expresión facial espontánea, también se asocia la dificultad en el discurso con daño frontal [57].

2) Lóbulo parietal. Se encuentra situado en la zona media del encéfalo. Esta zona permite la identificación de objetos y la comprensión de relaciones espaciales. Además interviene en el movimiento, la orientación, la interpretación del dolor y del tacto en el cuerpo [91].

Un daño en el lóbulo parietal izquierdo puede resultar en dificultad con la escritura (agrafia), dificultad con las matemáticas (acalculia), dificultad con el habla (afasia) y la inhabilidad de percibir objetos normalmente (agnosia). Daño en el lado derecho puede causar dificultad en la fabricación de cosas (apraxia construccional) y un déficit en la capacidad de dibujar. Un daño bilateral está caracterizado por la inhabilidad de integrar componentes de una escena visual (síndrome de Bálint) y una pérdida en la coordinación entre la visión y el movimiento de la mano (ataxia óptica) [59].

Si existe algún daño entre los lóbulos parietales y temporales, ocurren déficits principalmente en la memoria y la personalidad. Alguna lesión en el lóbulo parietal temporal izquierdo puede afectar la memoria verbal y la capacidad de recordar cadenas de dígitos. El lóbulo parietal temporal derecho se refiere a memoria no verbal y lesiones en esta región pueden producir cambios significativos en la personalidad [59].

3) Lóbulo occipital. Es la parte posterior del encéfalo e interviene en la visión [91]. Este lóbulo contiene la corteza visual, donde se proyectan en una representación geográfica las formas obtenidas en la retina [6].

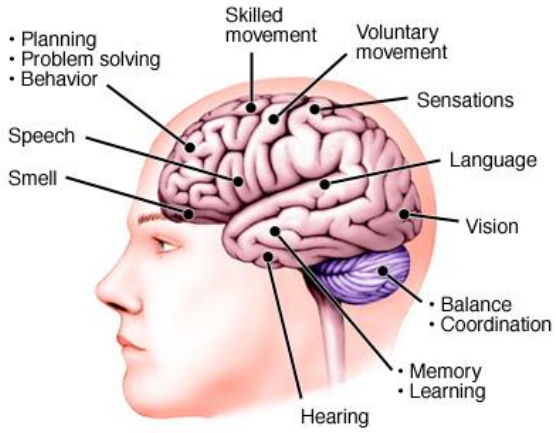
Este lóbulo es el centro del sistema visual de una persona. No es susceptible a daños debido a su ubicación dentro del cráneo, sin embargo, sí es posible sufrir alguna lesión en esta área. Cualquier trauma en esta región podría producir cambios sutiles al sistema visual-perceptivo, entre ellos se puede mencionar: alucinaciones visuales (percepción de imágenes visuales que realmente no existen), ilusiones visuales (distorsión de imágenes visuales) y pérdida de visión con exactamente el mismo campo cortado adentro de ambos ojos [58].

4) Lóbulo temporal. Se encuentran en las partes laterales del encéfalo. Este segmento interviene en la memoria, el habla, percepción y reconocimiento de estímulos auditivos y en cierto grado de reconocimiento de olfato [82].

La atención selectiva a la entrada visual o auditiva es común con daño a los lóbulos temporales. Las lesiones de las partes inferiores del lóbulo temporal conducen a agnosia visual. En la mayoría de personas el hemisferio izquierdo de este lóbulo es dominante en aspectos relacionados con la materia verbal y el hemisferio derecho está especializado en la elaboración de materia no verbal [77]. Daños en estas regiones también pueden resultar en efectos dramáticos en la personalidad del individuo [60].

En la Figura 5 se muestran qué áreas del cerebro son responsables para diferentes tareas o funciones que se realizan.

Figura 5. Funciones del cerebro [51].



C. ELECTROENCEFALOGRAFÍA

El cerebro se encuentra compuesto por millones de neuronas, las cuales transmiten información a través de potenciales de acción. Estos potenciales de acción son producidos por un flujo iónico entre los medios intra y extra-celular, generando una diferencia de potencial entre ellos que involucra un campo eléctrico. En 1924, dicho campo eléctrico fue medido por primera vez en el cerebro humano por el psiquiatra alemán Hans Berger [47].

Según la ubicación de los electrodos utilizados para la medición de la actividad eléctrica del cerebro, estas pueden clasificarse como invasivas (también conocidas como intra-craneales) o no invasivas. Las mediciones invasivas requieren la introducción de electrodos al cráneo, a través de un proceso quirúrgico. Dichas mediciones permiten la obtención de señales con amplitudes de 1 a 2 mV [47] y son conocidas como electro-corticografía (ECoG) si los electrodos se encuentran en la superficie de la corteza cerebral, o como estéreo electroencefalograma (E-EEG) si se encuentran en una región cerebral profunda [6]. Por su parte, las mediciones no invasivas utilizan electrodos colocados sobre el cuero cabelludo. Esto presenta la ventaja de ser menos costoso y de no implicar riesgos de daños cerebrales permanentes para el paciente. No obstante, las señales obtenidas tienen amplitudes de 100 µV [47], un valor menor al obtenido con los métodos invasivos, debido a que los potenciales eléctricos deben de atravesar el cráneo y cuero cabelludo [11], medios que no son tan buenos conductores eléctricos como el tejido cerebral y el líquido cefalorraquídeo [36]. A este segundo tipo de medición se le conoce como electroencefalografía (EEG). En EEG, los electrodos son normalmente de plata y miden aproximadamente 1 a 3 mm de diámetro [89]. Dichos electrodos son colocados sobre la cabeza de una persona de dos formas distintas: utilizando un gorro que se ajusta a la cabeza, como

muestra la Figura 6; o de manera individual. El gorro utiliza el sistema 10-20 para el posicionamiento de sus electrodos.

Figura 6. Gorro de electrodos para EEG [53].

Existen varias consideraciones a realizar en torno a las señales de EEG. Primero, se deben de revisar las consecuencias de su amplitud reducida. Una de ellas es que se deben de utilizar amplificadores para que la señal pueda ser procesada adecuadamente por otros circuitos [36]. También se hace necesaria la aplicación de sustancias conductoras (tales como solución salina, gel conductor o goma conductora) entre los electrodos y el cuero cabelludo, para evitar una mayor reducción en tal amplitud [36]. Asimismo, las señales de EEG sufren de ruido, el cual es introducido por diversas fuentes tales como señales eléctricas producidas por los músculos (i.e., señales de electromiografía) y radiación electromagnética producida por líneas de alimentación AC y diversos equipos (e.g., computadoras y televisiones) [74]. Debido a que las señales de EEG son, generalmente, de menor amplitud que el ruido recién mencionado, se sigue que las señales de EEG tienen una relación de señal a ruido reducida [74].

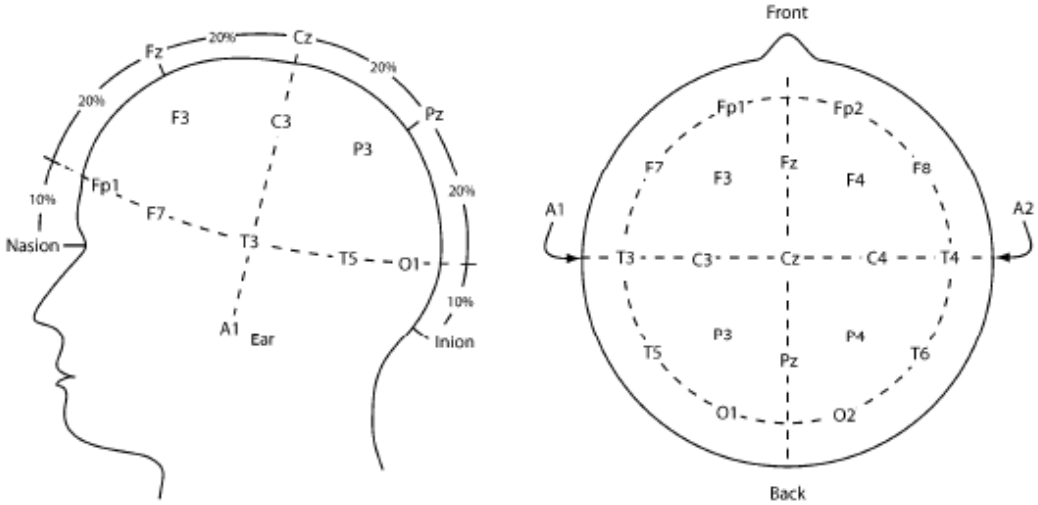
La segunda consideración gira en torno a la medición de las señales de EEG. Como primer punto, puesto que se están midiendo voltajes, se debe de tener un potencial de referencia con respecto al cual medir. Diferentes ubicaciones y métodos se han utilizado para obtener dicha referencia, tales como obtenerlas directamente de los electrodos de cierta ubicación específica (e.g., en la nariz), del promedio de dos electrodos (e.g., de los dos electrodos ubicados en los lóbulos de las orejas o en los mastoides) o del promedio de todos los electrodos disponibles [36]. Como segundo punto, se debe de conservar, de la mejor forma posible, la posición de los electrodos entre diferentes sesiones de medición, tanto para asegurar la reproductibilidad de los resultados como para no introducir más grados de libertad en la medición.

Con dicho propósito, se han creado diferentes estándares para la colocación de los electrodos sobre el cuero cabelludo, siendo uno de los más ampliamente utilizados el sistema internacional 10-20, ilustrado en

la Figura 7. Dicho sistema fue establecido por la federación internacional de electroencefalografía y consiste de la colocación de 21 electrodos sobre el cráneo de la persona. El nombre de 10-20 proviene del hecho de que las ubicaciones de los electrodos se encuentran separados por el 10% o 20% de la distancia, sobre el cráneo, entre el nasión y el inión [47]. El nombramiento de los electrodos se eligen según el área dónde se encuentran: F(frontal), C(Central), P(Parietal), T(Temporal) y O(Occipital). Para determinar si corresponden al hemisferio izquierdo o derecho, se utilizan números impares y pares, respectivamente [89].

Un punto adicional a considerar corresponde al espectro del EEG, el cual presenta componentes frecuenciales por debajo de 1 Hz hasta, aproximadamente, 50 Hz [47]. Esto implica que, en caso de muestrear una señal de EEG en el dominio del tiempo, esto se debe de realizar a una frecuencia de al menos 100 Hz, para poder cumplir con el teorema de muestreo de Nyquist-Shannon.

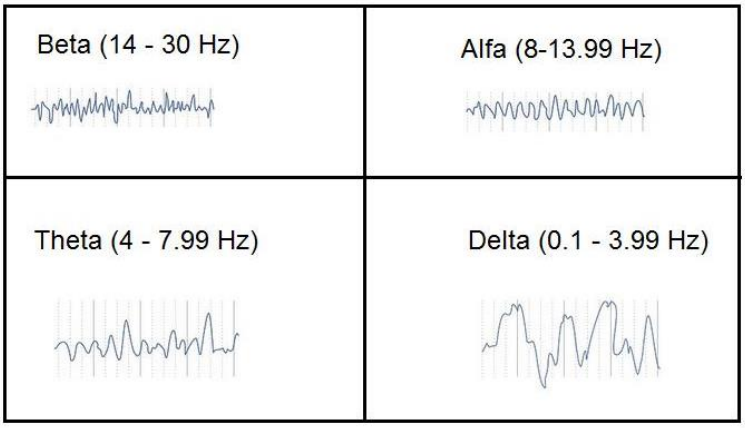
Figura 7**.** Sistema internacional 10-20 de ubicación de electrodos para EEG [75]**.**



Generalmente se han utilizado dos enfoques para analizar a las señales de EEG: las oscilaciones neuronales y los potenciales relacionados a eventos. Dichos enfoques se describirán a continuación, dando un mayor énfasis al segundo, por ser el de interés para este trabajo.

1. Oscilaciones neuronales. En este enfoque, se realiza una medición de EEG continua mientras el sujeto se encuentra quieto y sin realizar una acción motora específica. En dichos casos, las señales de EEG presentan un patrón oscilatorio que puede indicar el nivel de conciencia o estado psicológico del sujeto [73]. Las señales de EEG se clasifican en ondas delta, theta, alfa o beta, según la frecuencia de la oscilación, las cuales se ilustran en la Figura 8 y serán descritas a continuación.

Figura 8. Clasificación de ondas cerebrales [94].



a. Ondas Delta. Contienen a todas aquellas oscilaciones con frecuencia menor a los 4 Hz. En comparación con los otros tipos de ondas cerebrales, son las que poseen una amplitud de mayor magnitud (70 a 100μV a una distancia de 9 a 14mm) [75]. Dichas ondas son observadas cuando el sujeto se encuentra durmiendo, es un niño o padece de ciertas enfermedades del cerebro [90].

b. Ondas Theta. Estas ondas corresponden a oscilaciones que se encuentran entre los 4 y 8 Hz. Poseen una amplitud menor a las de las ondas Delta pero mayor a las demás (50μV a una distancia de 7mm) [75]. Pueden apreciarse en los lóbulos parietal y temporal de los cerebros de los niños, mientras que en los adultos, pueden ser causadas por estrés o frustración, así como pueden ser observadas durante algunas etapas del sueño [90].

c. Ondas Alfa. Las ondas Alfa exhiben una frecuencia de oscilación que se encuentra entre los 8 y 14 Hz. Poseen menor amplitud a las ondas theta (20 a 60μV a una distancia de 3 a 4mm) [75]. Tales ondas se presentan principalmente en el lóbulo occipital del cerebro, cuando una persona se encuentra despierta, relajada y con los ojos cerrados [90]. También se presentan en meditaciones y es estados de concentración profunda del sujeto [19].

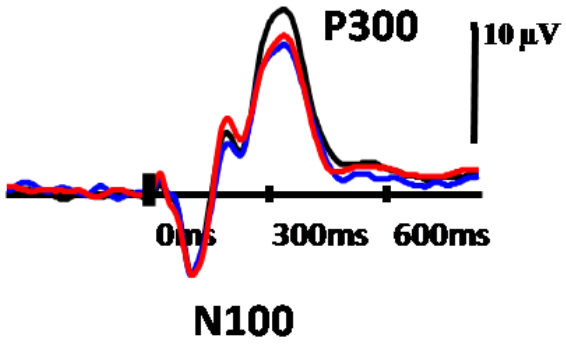
d. Ondas Beta. Dichas ondas corresponden a frecuencias de oscilación que se ubican entre los 14 y 30 Hz, las cuales se manifiestan principalmente en los lóbulos parietal y frontal. Además, poseen la menor amplitud de todas las ondas cerebrales (10 a 15μV a una distancia de 1 a 1.5mm) [75]. Las ondas Beta se dividen en dos grupos: Las Beta I, con frecuencias que varían entre los 13-20 Hz y las Beta II, cuyas frecuencias son superiores a los 20 Hz. Las ondas Beta I aparecen junto a las Alfa, mientras que las Beta II reflejan un estado de actividad mental intenso [90], asociado con una atención enfocada y un pensamiento activo vuelto hacia el exterior [98].

2. Potenciales relacionados a eventos. Conocidos en la literatura como ERP (por las siglas en inglés de *Event Related Potential*), son potenciales eléctricos generados en el cerebro como respuesta a un estímulo, ya sea interno o externo; e.g., podría ser generado por una experiencia sensorial (ya sea visual, auditiva o táctil) o psicológica [27]. En palabras de Picton, Lins y Scherg [69]:

> «Un ERP puede ser evocado por un estímulo externo o emitido por el cerebro mientras toma una decisión o inicia una respuesta. Los potenciales evocados se pueden clasificar también como exógenos o endógenos según si fueron provocados principalmente por las características físicas del estímulo o por sus efectos psicológicos.»

Los ERPs proveen información de cómo es procesado el estímulo, reflejan la respuesta del cerebro ante eventos en ambientes externos e internos al organismo. Estos potenciales se presentan como picos de voltaje en las señales EEG; los picos positivos se denotan con la letra P y los negativos con la letra N. Es importante mencionar que la amplitud de estos picos se encuentra en el rango de 5 a 10μV.

Figura 9. Ejemplo del componente P300 en un ERP [65].



La Figura 9 muestra dos ejemplos de ERP que se pueden presentar. Como se explicó anteriormente, la primera letra de la denotación indica si el pico es positivo o negativo. Además, el número a la par indica el tiempo (en ms) transcurrido entre el evento o estímulo y el pico [26]. En la figura anterior se puede observar que existen dos picos, uno negativo y otro positivo, y que ocurren 100ms y 300ms después del evento, respectivamente.

Debido a que la amplitud de estos potenciales es más pequeña que la del resto de componentes de una señal de EEG, estos no se pueden identificar sin utilizar algún tipo de procesamiento de la señal [88]. En este caso específico, estaremos interesados en un potencial provocado por un estímulo visual. A dicha clasificación de potenciales se les conoce como VEP, por las siglas en inglés de *Visual Evoked Potential.*

a. Onda P300. Uno de los ERPs más estudiados es la onda P300, el cual corresponde a una onda con un pico positivo que ocurre cierto tiempo después de que un estímulo objetivo y de baja probabilidad sea presentado al usuario dentro de una serie de estímulos irrelevantes [69]; i.e., para que ocurra la onda P300, el sujeto sobre el cual se realizan las mediciones de EEG debe de estar esperando un estímulo específico (al

cual se le llama estímulo objetivo) dentro de un conjunto de estímulos y, además, la probabilidad de que dicho estímulo objetivo ocurra debe de ser baja. En la literatura no se halló una fuente que defina qué se entiende como probabilidad baja, pero se han realizado estudios donde únicamente se contaban con cuatro posibles estímulos con probabilidad uniforme [11], por lo que una probabilidad de ocurrencia del 25% ya es adecuada para la aparición de esta onda.

La onda P300 se presenta, en la mayoría de sujetos, entre 300 y 500 ms luego de la presentación del estímulo objetivo, siendo el valor típico de 300 ms. El nombre P300 simboliza precisamente eso: Que el ERP se presenta como un pico positivo (de donde se obtiene la P en el nombre) con una latencia, o retraso, de 300 ms con respecto al estímulo (de donde se obtiene el 300) [69].

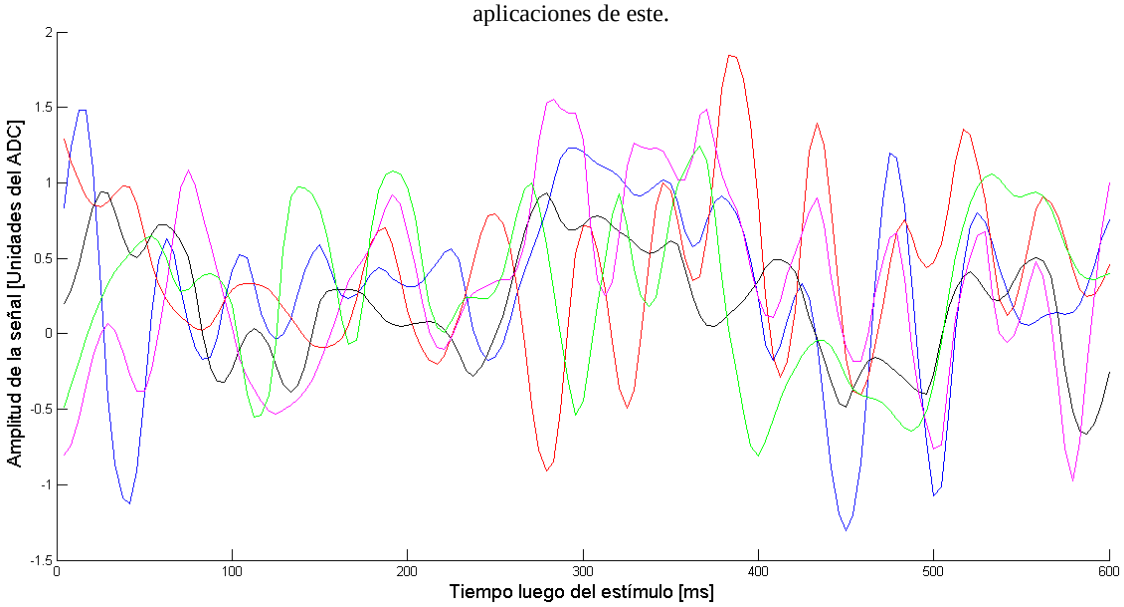
En cuanto al origen y la función de esta onda, este permanece sin conocerse totalmente, aunque se ha propuesto que está relacionado con el final de un proceso cognitivo, el cambio de la memoria luego de evaluar la información recibida o la transferencia de información de la conciencia [27].

Estudios en los cuales se han presentado dos estímulos de baja probabilidad, pero siendo solo uno de ellos un estímulo objetivo y el otro, uno distractor, han hallado la existencia de otra onda P300, i.e., de un segundo ERP que se presenta como un pico positivo con una latencia de 300 ms. Dicho ERP surge como respuesta al estímulo distractor y es conocido como un "P300 de novedad". A diferencia del P300 que surge como respuesta al estímulo objetivo y se aprecia con mayor amplitud en el lóbulo parietal, el P300 de novedad se concentra en las áreas frontal y central del cerebro [27]. Para diferenciarlos, se conoce como P3a al P300 de novedad, mientras que el P300 que ocurre como respuesta al estímulo objetivo se le ha dado el nombre de P3b. En este trabajo, a partir de este punto, al hablarse de la onda P300, se entenderá que se está hablando del P3b, asociado al estímulo objetivo, y no al P3a, asociado a un estímulo distractor.

Como ya se ha presentado anteriormente, la onda P300 se caracteriza por tener una amplitud pequeña y una relación de señal a ruido reducida. Esto se puede verificar al observar la Figura 10: En ella, se grafican en colores diferentes, señales de EEG grabadas sobre un mismo sujeto y adquiridas desde un mismo electrodo (específicamente, el electrodo ubicado en la posición Cz, la cual puede consultarse en la Figura 7) en los 600 ms posteriores a la presentación del mismo estímulo objetivo. Esto quiere decir que cada una de las señales graficadas debería de contener una onda P300. Sin embargo, se puede notar que no todas estas señales presentan un pico positivo visible entre los 300 y 500 ms transcurridos luego del estímulo. Inclusive es complicado hallar un patrón común entre todas estas señales, ya que cada una de ellas pareciera tener un comportamiento diferente, siendo el único elemento compartido la presencia de oscilaciones a, aproximadamente, 30 Hz. De acuerdo con lo presentando anteriormente, dichas oscilaciones corresponden a ondas Beta II, asociadas al estado de actividad mental intensa de una persona que se encuentra despierta. Esto constituye tan solo un ejemplo del resto de actividades neuronales que están ocurriendo dentro del cerebro y

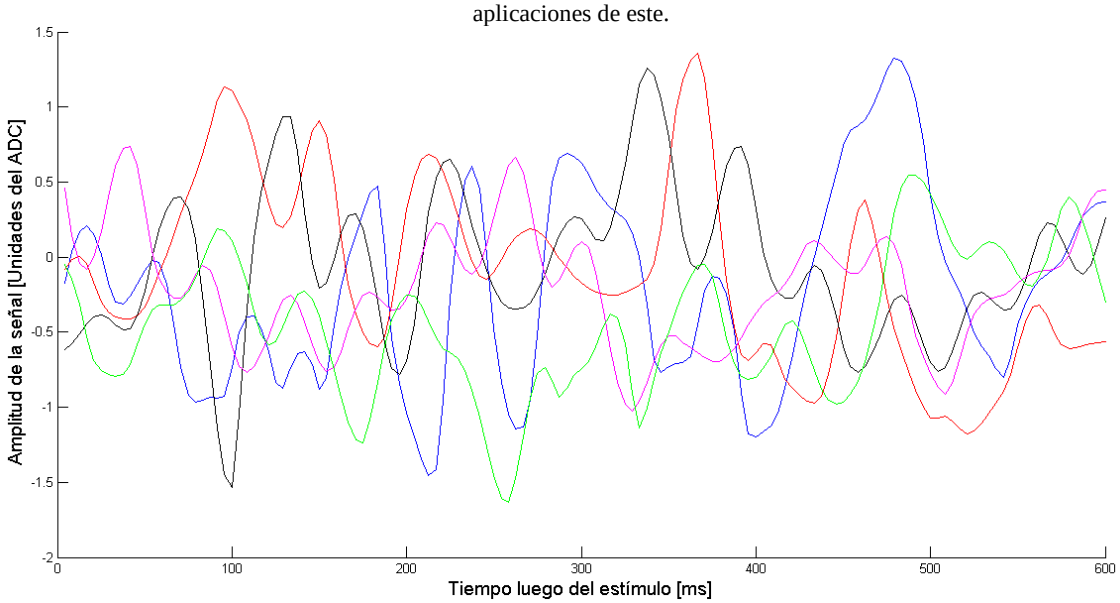
que generan sus propias señales de EEG, las cuales se sobreponen con el resto de señales generadas por otros procesos mentales y generan un ruido significativo para una aplicación en la cual se desea detectar únicamente a la onda P300.

Figura 10. Señales de EEG correspondientes a un P300
Estas señales fueron grabadas para un mismo sujeto, electrodo y estímulo objetivo presentado, pero en diferentes aplicaciones de este.



Para complementar el punto anterior, se presentan también las señales de EEG para el mismo sujeto y electrodo de las mostradas en la Figura 10, pero esta vez, se muestran los 600 ms posteriores a la presentación de un estímulo que no era el objetivo. Dichas señales pueden apreciarse en la Figura 11, donde se puede verificar que también presentan las oscilaciones observadas en la aplicación del estímulo objetivo y que es complicado hallar un patrón común entre ellas. Peor aún, algunas de estas señales exhiben características que las hacen similares a las señales mostradas en la Figura 10 (e.g., el pico positivo cercano a los 400 ms que se puede apreciar en las señales graficadas en color rojo en ambas figuras).

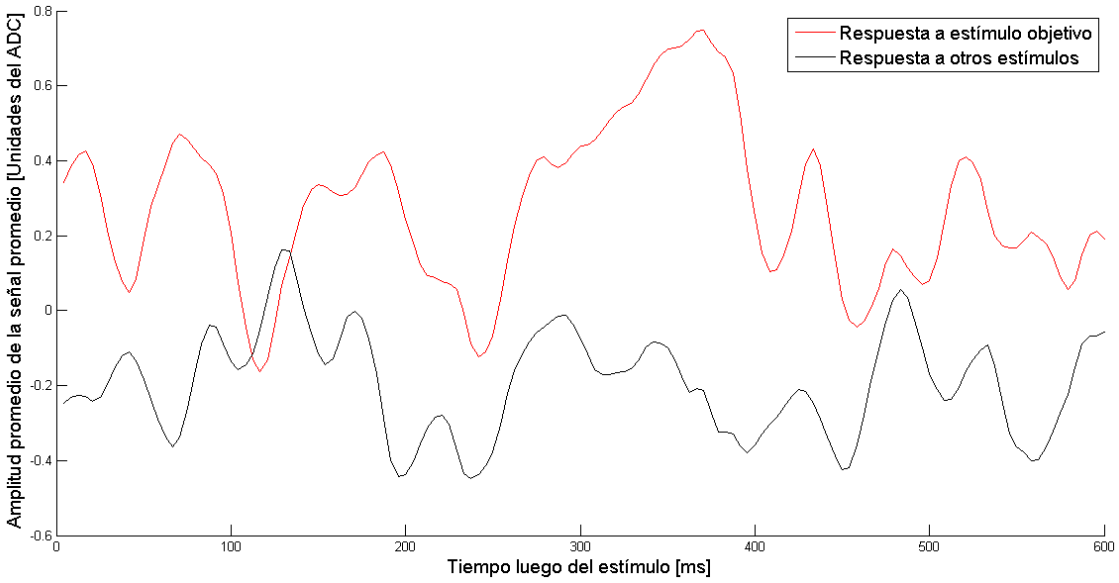
Figura 11. Señales de EEG no correspondientes a un P300
Estas señales fueron grabadas para un mismo sujeto, electrodo y estímulo no objetivo presentado, pero en diferentes aplicaciones de este.



Si deseamos clasificar las señales mostradas como aquellas que siguen a un estímulo objetivo y las que corresponden a un estímulo no objetivo, las características recién presentadas elevan la complejidad de la tarea de clasificación, debido a los siguientes dos puntos: Primero, es difícil hallar características compartidas entre las señales de la misma clasificación y, segundo, también es complicado hallar rasgos que permitan diferenciar a las señales de un tipo de las del otro. Por consiguiente, es necesario primero procesar las señales para mejorar su relación de señal a ruido y así, facilitar la tarea de clasificación.

En el caso de la detección de la onda de P300, uno de los métodos más utilizados para mejorar la relación de señal a ruido es el de promediar múltiples mediciones de EEG realizadas sobre el mismo estímulo [27]. La idea detrás de esto es que el resto de la actividad cerebral y otras fuentes de ruido en la señal se cancelarán durante el promedio, mientras que lo único constante en la señal, que debería de ser el ERP del P300, se conservará. Para ilustrar lo anterior, se presenta la Figura 12, en la cual se muestra, en color rojo, el promedio de las señales ilustradas en la Figura 10 con otras 10 señales más correspondientes al mismo sujeto, electrodo y estímulo, mientras que en color negro se grafica el promedio de las señales de la Figura 11 junto a otras 10 señales propias del mismo sujeto, electrodo y estímulo. Ya con esta etapa de la aplicación del promedio, se puede identificar de forma más clara la presencia de la onda P300 en respuesta a estímulos objetivos y su ausencia en respuesta a estímulos no objetivos, como se muestra en la Figura 12. Asimismo, se puede apreciar una reducción en las oscilaciones dentro de las señales.

Figura 12. Señales de EEG promediadas
Las señales promediadas corresponden a un mismo sujeto, electrodo y estímulo.

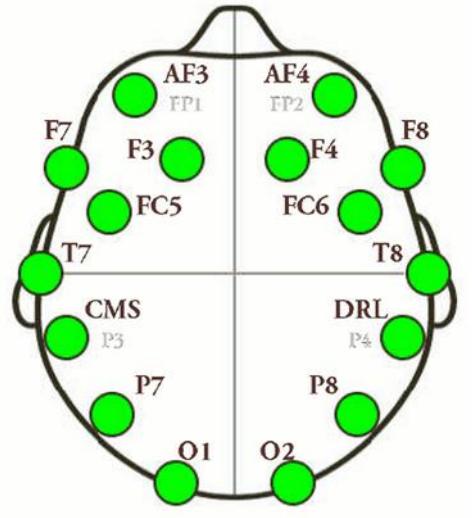


A pesar de que el promedio ofrece una técnica propicia para obtener la señal correspondiente a la onda P300, las características de esta misma, tales como su latencia y amplitud, varían entre personas e incluso, dentro de diferentes promedios para una misma persona. Esto implica que la tarea de clasificación sigue siendo una compleja, puesto que no se puede definir un tiempo de ocurrencia del pico del P300, ni una amplitud, que sea universal y funcione para todos los sujetos. Por consiguiente, generalmente se utilizan algoritmos de aprendizaje automático para detectar las características que diferencian a cada uno de los tipos de señales según el sujeto [27] y lograr una clasificación adecuada.

3. Emotiv EPOC. Existen varios equipos utilizados para captar las señales eléctricas cerebrales. Un ejemplo de ellos es el Emotiv EPOC *Neuro-Headset* (Figura 13) desarrollado por Emotiv *Systems*. El Emotiv EPOC utiliza el sistema 10-20 de posicionamiento, posee 14 canales EEG y 2 referencias (CMS y DRL en la Figura 14).

Figura 13. Emotiv Epoc *Neuro-Headset* [22].

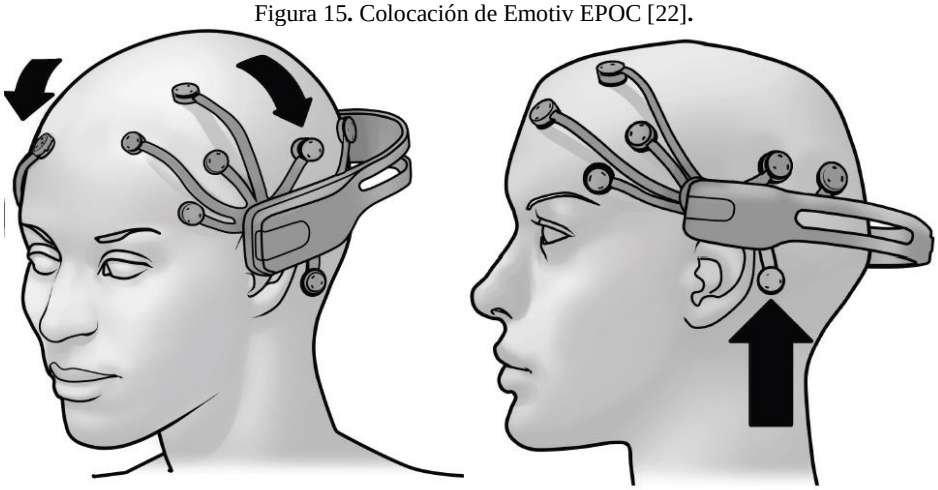
Figura 14. Posicionamiento de Electrodos en Emotiv EPOC [22].



El Emotiv EPOC muestrea cada canal en forma secuencial con una frecuencia de 128 Hz, por lo que utiliza un muestreador interno de 2048Hz. El ADC que utiliza posee una resolución de 14 bits (1 LSB equivale a 0.51μV) e incluye internamente un filtrado de 0.16 a 43 Hz, además de un filtro quita banda de 50 y 60 Hz. Utiliza un filtro pasa bajos de orden 5 para quitar el ruido de las señales biopotenciales [22].

Además de los canales EEG, el EPOC incluye un giroscopio para posicionamiento de la cabeza y se comunica de forma inalámbrica con la computadora, utilizando *bluetooth*. Los electrodos utilizados para este gorro deben ser humedecidos con solución salina antes de su colocación sobre el casco. Posteriormente se deben enroscar cada uno de los electrodos en el casco y colocarlo en la cabeza del usuario como muestra la siguiente figura:

Figura 15. Colocación de Emotiv EPOC [22].



El programa que ofrece Emotiv *Systems* proporciona una interfaz gráfica, llamada *EmoEngine*, la cual permite determinar la calidad de la señal en cada uno de los electrodos. Para esto aparece una figura similar a la Figura 14 y mediante un código de colores se determina qué electrodos aún no poseen buena señal. Los códigos son: verde, señal ideal; amarillo, señal regular; anaranjado, señal pobre; rojo, señal muy pobre y negro, no hay señal [22].

D. ALGORITMOS DE APRENDIZAJE AUTOMÁTICO

De acuerdo con Andrew Ng [61], una de sus primeras definiciones de aprendizaje automático (o *machine learning*, en inglés) fue dada en 1959 por Arthur Samuel y es la siguiente:

> «Es un campo de estudio que otorga a las computadoras la habilidad de aprender sin ser explícitamente programadas.»

Ng [61] también presenta otra definición de lo que es el aprendizaje automático, dada por Tom Mitchell en 1998, la cual se enfoca más en el aprendizaje en sí:

> «Se dice que un programa de computadora aprende de la experiencia E con respecto a una tarea T y alguna medida de desempeño P si su desempeño en T, dado por P, incrementa con la experiencia E.»

Las dos definiciones anteriores dejan claro que los algoritmos de aprendizaje automático son aquellos que permiten que una computadora sea capaz de, como dice su nombre, "aprender", entendiendo dicha palabra como mejorar su desempeño en una tarea específica. Debe de recalcarse que se está hablando de una tarea específica: i.e., el algoritmo no es capaz de ofrecer a la computadora de un aprendizaje completo y general que le permite tratar con todas las situaciones que se le presenten, sino que estará entrenado específicamente para una tarea concreta. Además, para lograr dicha mejora en el desempeño, la computadora requiere de información previa respecto a la tarea asignada; e.g., un algoritmo de aprendizaje automático puede ser capaz de identificar los números escritos a mano por diferentes personas si se le entregan como entradas varios ejemplos de dichos números. Posteriormente, dicho algoritmo será capaz de reconocer los números escritos por varias personas, mas no será capaz de reconocer letras escritas a mano, puesto que esa no fue la tarea para la cual fue entrenado. También puede ocurrir que no reconozca los números escritos por ciertas personas, en cuyo caso tendría que darse más experiencia al algoritmo, a través de nuevos ejemplos, para poder mejorar su desempeño y, así, reconocer los números escritos a mano que antes no podía identificar.

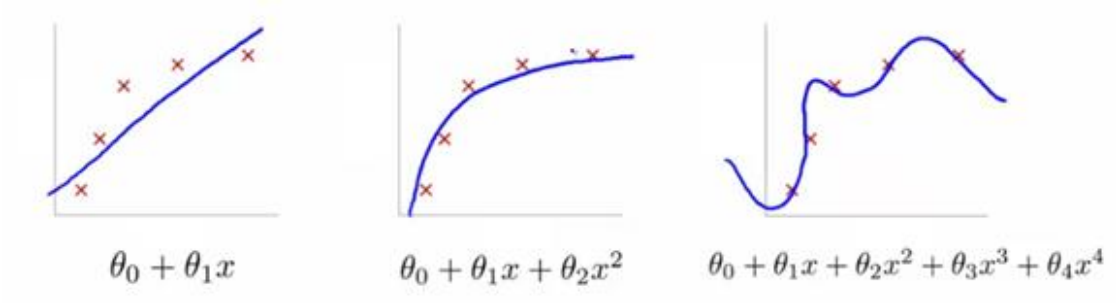
Los algoritmos de aprendizaje automático pueden dividirse en dos grandes clasificaciones: Los de aprendizaje supervisado y los de aprendizaje no supervisado. En la primera clasificación, los datos de ejemplo con los cuales se alimenta al algoritmo ya tienen una etiqueta que los identifica [43]: e.g., siguiendo el caso de los números anteriormente presentados, cuando se entregan los datos de entrenamiento (digamos, una imagen) al algoritmo, tal información está acompañada de a cuál número corresponde cada imagen. Entonces, el algoritmo se entrenará sobre dichos datos y etiquetas para que, la próxima vez que se le ingrese la imagen

de una escritura a mano del número cero, el programa sea capaz de identificar que tal imagen corresponde a la categoría de "número cero". Por su parte, los algoritmos no supervisados reciben datos que no se encuentran etiquetados y es el algoritmo quien se encarga de agruparlos según algún criterio [43]. La utilidad de estos algoritmos radica en, como lo indica Kotsiantis [43], descubrir formas de clasificar a los datos que son útiles, pero que no pueden ser fácilmente observadas.

En este caso, estaremos interesados en los algoritmos de clasificación supervisada, puesto que, como se explicará más adelante, contaremos con varias señales de EEG que ya se encuentran identificadas como parte de una de dos categorías: Presenta o no a la onda P300; i.e., alimentaremos al algoritmo de aprendizaje automático con ejemplos de señales ya etiquetadas para que cree un modelo con el cual luego podrá, al recibir nuevas señales de EEG, clasificarla como una señal que contiene, o no, a la onda P300.

Desde una perspectiva general, para lograr su cometido, los algoritmos de clasificación supervisada utilizan una función costo, la cual cuantifica el error de la clasificación realizada a partir de un modelo (correspondiente a una función matemática) sobre los datos recibidos como entrada. El algoritmo de aprendizaje automático se encarga entonces, de una forma iterativa, de modificar los parámetros de su modelo de clasificación con el objetivo de reducir su función costo (i.e., reducir el número de errores en la clasificación realizada sobre los datos de entrada). En dicho escenario puede ocurrir que el algoritmo de clasificación minimice tanto su función costo que el modelo de clasificación se encuentra sobre-ajustado (en inglés, *over fitted*) a los datos de entrenamiento recibidos y que, por consiguiente, no se ajuste a pequeñas variaciones en nuevos datos de entrada, desembocando en varias clasificaciones erróneas. Para resolver este problema, se utiliza la regularización, la cual consiste en añadir un término a la función costo que evite que el modelo matemático se sobre-ajuste [61]. Ng [61] indica que otro cuidado que se debe de tener al momento de implementar un algoritmo de aprendizaje automático está en la desviación (en inglés, *bias*): Esto puede ocurrir si el modelo de clasificación utilizado no presenta la complejidad suficiente como para capturar las diferencias entre los datos: e.g., supongamos que se desean diferenciar los elementos en una imagen que se encuentran dentro de un círculo de aquellos que no. En dicho caso, si deseamos separar ambos grupos utilizando únicamente una línea recta, nuestra clasificación será una muy pobre, por lo que se dice que sufre de una alta desviación que produce un sub-ajuste (*under fitting* en inglés).

Figura 16. Sub-ajuste y sobre-ajuste [61].
Ejemplos de regresiones que presentan sub-ajuste (izquierda) y sobre-ajuste (derecha), así como una que no exhibe dichos problemas (centro).

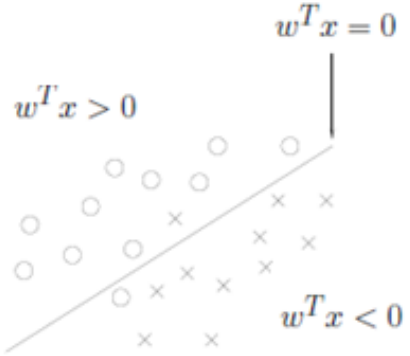


Para aclarar los conceptos anteriores de sobre y sub-ajuste, se puede observar la Figura 16, donde se aplican dichos conceptos en el contexto de las regresiones. En dicha figura, se grafican tres sistemas de coordenadas, cada uno con datos conocidos (mostrados como cruces rojas) y modelos de regresión obtenidos para ellos (trazados en azul). El modelo presentado en el extremo izquierdo sufre de desviación y sub-ajuste, ya que la cantidad de parámetros en él (que son dos, θ_0 y θ_1, por tratarse de un modelo lineal) no son suficientes para capturar la variación en los datos. Por el contrario, la regresión del extremo izquierdo sufre de una gran variación, ya que presenta un comportamiento distinto al de la tendencia presentada por los datos. La razón de ser de esa alta variación es que el modelo busca pasar por todos los datos que se le han dado, por lo que está sobre-ajustado a ellos: Si bien la regresión (o en el caso de los algoritmos de aprendizaje automático, la predicción) será exacta para los datos de entrenamiento, las predicciones para nuevos datos serán desacertadas, puesto que el modelo no captura la tendencia en los datos. Finalmente, la regresión que se encuentra en el centro no sufre de ninguno de los dos problemas: Si bien, no pasa a cabalidad por todos los datos de entrada, conserva la tendencia presentada por los datos, por lo que las predicciones realizadas para elementos fuera del conjunto de entrada serán adecuadas.

Como siguiente punto, se presentarán algunos de los algoritmos de clasificación supervisada más utilizados al momento de identificar a la onda P300, de acuerdo con Manyakov [48], Krusienski [44] y Lotte [46]. En dichas descripciones, definiremos a X como una matriz que contiene a las características de los elementos a clasificar, correspondiendo cada fila de la matriz a un elemento diferente y cada columna a una característica distinta (incluyendo una columna cuyos elementos serán todos uno, lo cual servirá para poder añadir un término constante durante la clasificación), mientras que y es un vector columna cuyas filas corresponden a la etiqueta de cada uno de los elementos que se están dando al algoritmo para entrenar. Asimismo, x corresponderá a un vector columna que contiene las características de un elemento, siendo su primer elemento un uno (para añadir el término constante); i.e., la matriz X está compuesta por varios vectores x transpuestos, uno por cada uno de los elementos utilizados para entrenar al algoritmo.

1. Análisis de discriminante lineal de Fisher. También es conocido como FLDA, de las siglas en inglés de *Fisher's Linear Discriminant Analysis.* Este algoritmo se encarga de separar cada una de las clasificaciones utilizando híper-planos; en el caso de una clasificación binaria, el grupo al cual cada uno de los elementos pertenece vendrá dado por el lado del híper-plano en el cual dicho elemento se encuentra [46]. Esto se ilustra en la Figura 17.

Figura 17. Híper-plano hallado por FLDA para separar a dos clases [46].



De acuerdo con Krusienski [44], para clasificaciones binarias donde ambas clases son gaussianas y con la misma covarianza, las soluciones de este análisis son equivalentes a la de una regresión lineal de mínimos cuadrados ordinarios. Esto quiere decir que para determinar los coeficientes del modelo de clasificación, basta con computar:

$$w = (X^T X)^{-1} X^T y$$

Y para obtener la clasificación correspondiente $\hat{y}$ para un elemento x, solamente se debe de evaluar:

$$\hat{y} = w^T x$$

Y observar si el resultado $\hat{y}$ es mayor o menor que cero, como se indica en la Figura 17.

La ventaja de este método es que presenta una complejidad computacional reducida, pero presenta el inconveniente de ser lineal, por lo que podría no clasificar adecuadamente a aquellos datos que presentan una relación no lineal [46].

a. Análisis de discriminante lineal *Stepwise*. Recibe el nombre de SWLDA, por las siglas en inglés de *Stepwise Linear Discriminant Analysis* y es una derivación del FLDA, en la cual se toman en consideración para la clasificación únicamente a aquellas características que representan una mayor cantidad de información [27]. Krusienski [44] describe el mecanismo que permite la selección de las características a utilizar:

> «Se inicia con ninguna de las características en la función discriminante y luego se añade a dicha función una característica estadísticamente significativa para predecir correctamente la clasificación (la cual tiene un valor p menor a 0.1). Se prosigue de esta manera y cada vez que se añade una nueva característica a la función discriminante, se retiran a aquellas que tienen un valor p mayor a 0.15, lo cual indica que ha perdido la significancia estadística para la predicción. Se repite este proceso hasta

llegar a un número máximo de características seleccionadas o hasta que no existan características que satisfagan los criterios de inclusión y exclusión de la selección.»

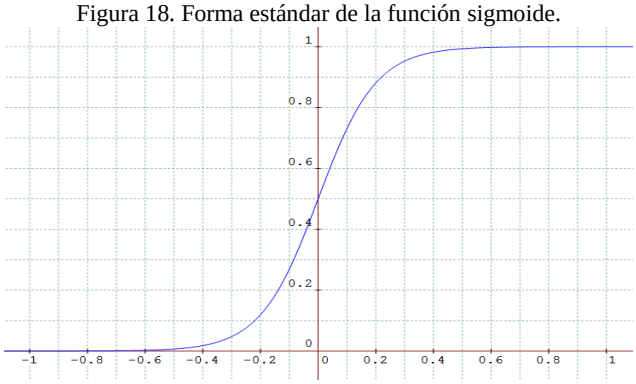
Los límites de los valores p que permiten que una característica sea incluida o rechazada de la función discriminante pueden ser variados, pero los valores recién mostrados han sido los utilizados en múltiples estudios relacionados con la onda P300 [48]. La ventaja que presenta el SWLDA es que reduce el número de características a considerar en la clasificación.

2. Regresión logística. La regresión logística, o *logistic regression*, es un algoritmo de clasificación discreta, que permite clasificar de manera discreta (0 o 1) y, como se observa en la Figura 18, no sobrepasa estos valores, situación que podría ocurrir en una regresión lineal. Para ello, se utiliza la función sigmoide, la cual viene dada por la siguiente ecuación:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Esta función tiene la ventaja que provee como resultado la probabilidad de que la hipótesis (clasificación) sea igual a 1. Siguiendo la notación utilizada para el FLDA y siendo $h(x)$ el resultado la clasificación para el vector x, se observa que para hacer de la salida de la función sigmoide una binaria, se puede clasificar $h(w^T x) = 1$ si $w^T x \geq 0$ y $h(w^T x) = 0$ en caso contrario [61].

Figura 18. Forma estándar de la función sigmoide.



Si se utiliza únicamente una característica dentro del vector x, las regresiones logísticas terminarán siendo lineales. Para obtener curvas de mayor grado o complejidad, se puede utilizar el mapeo de características (en inglés, *feature mapping*), el cual consiste en elevar los grados de los atributos y agregarlos a los arreglos iniciales. Por ejemplo, si se tienen atributos x_1 y x_2, se podrían agregar, e.g., x_1^2, x_2^2 o $x_1 x_2$, como atributos. Esto resultaría en regresiones más complejas, sin embargo, el procesamiento sería más tardado [61].

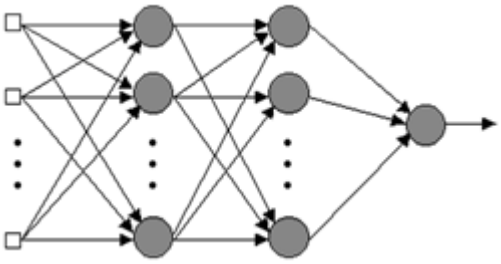
A pesar de que el FLDA y la regresión logística permiten, en principio, solamente una clasificación binaria, puede utilizarse para realizar una clasificación multi-clase a través de la técnica de uno contra todos. En dicha técnica, para clasificar a N distintas clases se utilizan N diferentes clasificadores. Al entrenar a cada uno de los clasificadores, se selecciona una de las clases, cuya etiqueta será convertida a un “1”, mientras

que las demás etiquetas serán colocadas como “0”. Así, cada uno de los clasificadores podrá identificar si un elemento pertenece a una clase específica o no, y observando las salidas de los N clasificadores, se puede determinar la clase exacta a la cual pertenece [61].

3. Red neuronal artificial. Conocido en inglés como *artificial neural networks* (ANN), este algoritmo propone un arreglo de elementos, llamados neuronas artificiales, los cuales cuentan con una función que dicta su salida según las entradas que tiene y a partir de las cuales se pueden lograr clasificaciones no lineales [48]. Generalmente, la función de activación de las neuronas artificiales es la función sigmoide, previamente presentada, donde x es la sumatoria ponderada de las entradas a la neurona y $f(x)$, su salida [61].

Una de las redes neurales más utilizadas es la *Multilayer Perceptron*, la cual se encuentra compuesta por una capa de entradas (una entrada por cada una de las características), una o varias capas ocultas y una capa de salida (conformada, en el caso de una clasificación binaria, por una neurona). Dicha red neuronal se ilustra en la Figura 19 y se caracteriza por estar conectadas todas las salidas de una capa a las entradas de cada una de las neuronas de la siguiente capa. Entonces, de lo que se encarga el algoritmo de aprendizaje automático es de fijar los pesos que tendrá cada una de las entradas para las diferentes neuronas, de tal forma que la clasificación final sea adecuada [61].

Figura 19. Estructura de una red neuronal *Multilayer Perceptron* [56].

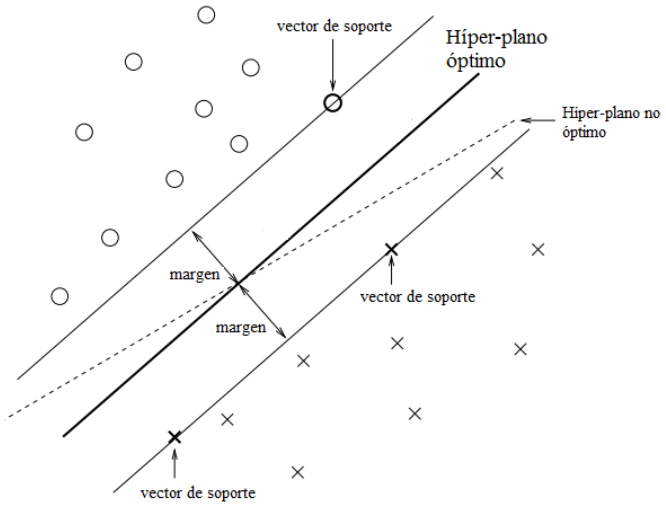


Durante el aprendizaje de la red neural, se determina el error de cada salida y se asigna un factor de aprendizaje. Con el error de cada neurona y el factor de aprendizaje se crea un gradiente, con el cual se ajustan los pesos de cada nodo, al realizar múltiples iteraciones. Debido a que la salida de cada neurona afecta a las siguientes capas, se realiza un “*Back Propagation*”, el cual ajusta los pesos de las capas anteriores según el error y el peso da cada neurona [71].

De tener suficientes neuronas y capas, la red neuronal artificial tiene la capacidad de aproximar a cualquier función continua [48]. Esto dota al clasificador de una gran versatilidad. Sin embargo, si se utilizan muchas neuronas y capas, el algoritmo puede fácilmente presentar problemas de sobre-ajuste [48]. Asimismo, en el caso en que no se tengan capas ocultas, sino solo las de entrada y de salida, la red neuronal se llama únicamente *Perceptron* y es equivalente a un FLDA [46].

4. Máquina de vectores de soporte. Conocido también como SVM (del inglés *Support Vector Machine*), este algoritmo es similar al FLDA en el sentido en que separa las clasificaciones utilizando un híper-plano. Sin embargo, a diferencia del FLDA, el criterio para formar dicho híper-plano consiste en que éste presente la mayor separación entre márgenes, i.e., la mayor distancia a los datos de entrenamiento más cercanos [46], como se ilustra en la Figura 20.

Figura 20. Híper-plano que separa a las dos clasificaciones en un SVM [46].



Con el objetivo de evitar sobre-ajuste, se agregó en el algoritmo un término de regularización C. Incrementar dicho término aumentará la variación de la clasificación, lo que podría provocar un sobre-ajuste, mientras que disminuirlo aumenta la desviación, lo que podría terminar desembocando en un sub-ajuste [61].

Lo descrito anteriormente corresponde a un SVM con *kernel* lineal. Sin embargo, también existen SVMs con otros *kernels*, siendo uno de los más utilizados el SVM de *kernel* gaussiano, el cual permite realizar clasificaciones no lineales. En dicho caso, cada uno de los puntos de entrada se mapea utilizando la siguiente función:

$$f_i = e^{-\frac{\left|\left|x-l^{(i)}\right|\right|^2}{2\sigma^2}}$$

Donde x corresponde a un punto de entrada; $l^{(i)}$, es uno de los puntos del conjunto de entrenamiento dado inicialmente al algoritmo; σ, es un parámetro del mapeo y f_i es el resultado del mapeo. Al analizar de forma intuitiva a esta expresión, se puede observar que lo que se está haciendo es obtener un valor que depende del cuadrado de la distancia entre un punto desconocido y uno conocido previamente del entrenamiento [61]. De hecho, mientras más cercano se esté a uno de los puntos de entrenamiento, el valor de dicha función aproxima a uno, mientras que, si se está más alejado, la función aproxima a cero. La velocidad con que dicha función decrece se puede controlar utilizando el parámetro σ: Si dicho valor es muy grande, f_i decrece lentamente, por lo que se consideran distancias más grandes como cercanas y se puede llegar a sufrir de sub-ajuste. Por

el otro lado, si σ es un valor pequeño, f_i decrece rápidamente y se consideran únicamente a las distancias pequeñas como similares con los datos de entrenamiento, favoreciendo el sobre-ajuste. Por consiguiente, para evitar que los SVMs de *kernel* gaussiano exhiban sub o sobre-ajuste, es necesario elegir unos valores de C y σ adecuados.

De acuerdo con Lotte [46], los SVMs presentan la ventaja de tener buenas propiedades de generalización, de ser difíciles de sobre-ajustar y de ser insensibles a la maldición de la dimensión. Esto último corresponde a un fenómeno que ocurre en los algoritmos de aprendizaje automático al tener muchas más características que elementos de entrenamiento. Esto produce resultados de clasificación pobres puesto que, como lo expresa Lotte [46]:

> «La cantidad de datos necesarios para describir adecuadamente a las diferentes clases aumenta exponencialmente con la dimensión de los vectores de características.»

5. Métricas de desempeño en algoritmos de aprendizaje automático. Luego de entrenar a un algoritmo de aprendizaje automático, hace falta una métrica que permita cuantificar la calidad de las predicciones realizadas por este mismo. Además, como ya se presentó anteriormente, algunos algoritmos requieren la definición de ciertos parámetros (e.g., C y σ en el caso de un SVM de *kernel* gaussiano), cuyos valores no se conocen de antemano ni pueden ser determinados a partir de una fórmula matemática, sino que estos deben de ser sintonizados por medio de un método de prueba y error hasta hallar los valores que optimizan el desempeño del clasificador.

Una de las métricas generalmente utilizadas en los algoritmos de aprendizaje de máquinas es la exactitud de la predicción: i.e., de todas las predicciones realizadas, cuántas resultaron ser correctas, número que se tiende a expresar como porcentaje. Sin embargo, se debe de ser cuidadoso al momento de obtener dicha métrica: e.g., si la métrica se obtiene sobre los datos utilizados para el entrenamiento del algoritmo, ya que este modificó los parámetros de su clasificador para optimizar la agrupación de dichos datos, se obtendrán exactitudes cercanas al 100%. Esto no implica que el algoritmo pueda realizar predicciones correctas sobre datos distintos a aquellos de con los cuáles se entrenó. Por consiguiente, para obtener una métrica válida, esta debe de ser medida dentro de un conjunto de datos que no pertenecen al conjunto de entrenamiento, por lo que son desconocidos para el algoritmo de clasificación y, por ende, este último no se encuentra ajustado específicamente para tales datos.

De manera similar, para el caso en que se deben de definir ciertos parámetros para el algoritmo, las métricas que permitirán decidir los valores óptimos de los parámetros deben de ser computadas sobre un conjunto de datos distinto a aquel con el cual se entrenó al algoritmo. Posteriormente, una vez ya estén elegidos los valores de tales parámetros, para reportar una métrica propia del clasificador resultante, se debe de utilizar un tercer conjunto de datos que no hayan sido utilizados en el entrenamiento del algoritmo ni en

la sintonización de sus parámetros, ya que de lo contrario se caería de nuevo en probar la clasificación con elementos conocidos para los cuales el programa se encuentra ya preparado.

Es por las razones anteriores que Ng [61] recomienda que, de los datos disponibles, se dividan estos en tres conjuntos: Uno de entrenamiento, un segundo de validación cruzada y un tercero de prueba. El primer conjunto será utilizado para entrenar al algoritmo clasificador, el segundo servirá para elegir la mejor combinación de parámetros del algoritmo clasificador y el tercero para obtener una métrica final del desempeño del clasificador. Ng [61] además recomienda que las proporciones para dichos conjuntos sean del 60%, 20% y 20%, respectivamente, del total de datos disponibles para preparar al clasificador.

Por otra parte, en dado caso se desee detectar si el algoritmo sufre de problemas de variación (sobre-ajuste) o desviación (sub-ajuste), Ng [61] indica que una técnica adecuada consiste en realizar un gráfico del error (el cual se puede calcular como 100% menos la exactitud) contra el número de elementos que conforman al conjunto de entrenamiento, tanto para el conjunto de entrenamiento como para el conjunto de validación cruzada. Es de esperar que, mientras aumenta el número de elementos del conjunto de entrenamiento, el error sobre el conjunto de entrenamiento incrementa (puesto que cada vez se tendrán más elementos a considerar en la clasificación) y el error sobre el conjunto de validación disminuye (puesto que el clasificador captura de mejor manera la variación entre los datos de cada una de las clasificaciones mientras más datos de ambas clasificaciones conoce). En dicha situación, puede ocurrir uno de tres escenarios: El primero de ellos consiste en que ambas curvas de error se estabilizan en un mismo valor, pero dicho valor corresponde a un alto error. En este caso, se detecta que el algoritmo está sufriendo de un sub-ajuste y que deben de añadirse más características (las cuales podrían inclusive ser potencias de las ya presentes) para reducir el error. En estos casos, aumentar el conjunto de entrenamiento no ayudará a reducir el error [61]. En el segundo escenario, las curvas de error terminan en valores de error significativamente distintos. En dicho caso, se detecta que el algoritmo está sufriendo de sobre-ajuste, lo cual se expresa como un pequeño error sobre el conjunto de entrenamiento pero un gran error en el de validación cruzada. En estos casos, incrementar el tamaño del conjunto de entrenamiento ayuda a reducir el sobre-ajuste [61]. Finalmente, la tercera situación que puede ocurrir es que ambas curvas de error lleguen a un mismo valor que corresponde a un error pequeño. Esto quiere decir que el algoritmo proporciona una clasificación adecuada para el conjunto de entrenamiento y, más importante aún, para el de validación cruzada, verificando el adecuado funcionamiento del mismo.

Hasta ahora, se ha hablado únicamente de la exactitud y del error como métricas de desempeño: Se desea que la métrica de exactitud sea lo más cercana al 100% como sea posible, mientras que se busca que el error se acerque al 0%. Sin embargo, cuando se tiene una clasificación binaria sesgada, esta métrica podría ofrecer una percepción inadecuada del desempeño del clasificador [61]. Al hablar de clasificación binaria sesgada, se está haciendo referencia a una clasificación que únicamente cuenta con dos posibilidades, pero cuya frecuencia de ocurrencia no es uniforme: i.e., uno de los eventos tiene una mayor ocurrencia que el otro. Para

ilustrar esto, consideremos un algoritmo de aprendizaje automático que, a partir de sus entradas, desea determinar si el elemento a clasificar pertenece a la clase A o B, considerando que la clase B solo ocurre en el 1% de la población, es complicada de detectar y es la clase que nos interesa. En este escenario es posible que, al recibir los datos del conjunto de entrenamiento y ser evaluado con los datos del conjunto de prueba, se obtenga una exactitud del 99% para el clasificador. Si tan solo observamos el número de exactitud, sin considerar el resto de la situación, la métrica nos indica que el clasificador es satisfactorio. Pero si luego analizamos las salidas individuales, podríamos hallar que el clasificador está prediciendo únicamente la clase A, la cual ocurre el 99% de las veces y es de ahí de donde obtiene semejante métrica para la exactitud: i.e., el clasificador realmente no está realizando ningún trabajo útil, puesto que nunca es capaz de identificar correctamente a un elemento de la clase B.

Para estas clasificaciones binarias sesgadas, deben de utilizarse otras métricas que reflejen la capacidad del clasificador de identificar la clase de interés que ocurre con baja frecuencia: la precisión y la exhaustividad. Antes de presentar sus definiciones, se debe de aclarar que entenderemos como un elemento positivo a aquel que pertenece a la clase B del ejemplo anterior; i.e., que tiene un baja frecuencia de ocurrencia y es la clase de interés en la clasificación. El resto de los elementos serán parte de la clase negativa. Además, debemos de definir cuatro términos más: Verdaderos positivos corresponden a aquellos elementos de la clase positiva que fueron etiquetados por el algoritmo como tales; falsos positivos son aquellos elementos que fueron etiquetados por el algoritmo como pertenecientes a la clase positiva, cuando en realidad eran parte de la negativa; falsos negativos, si fueron etiquetados como miembros de la clase negativa y pertenecían a la positiva; y verdaderos negativos si son elementos de la clase negativa que fueron etiquetados por el algoritmo correctamente. La cantidad de elementos pertenecientes a cada una de estas definiciones suelen representarse en una matriz, llamada la matriz de confusión, la cual se presenta en el Cuadro 1.

Cuadro 1. Matriz de confusión.

Etiqueta real\Etiqueta predicha	Clase positiva	Clase negativa
Clase positiva	Verdadero positivo	Falso negativo
Clase negativa	Falso positivo	Verdadero negativo

De acuerdo con Ng [61], la precisión es igual al número de verdaderos positivos sobre el total de elementos predichos como positivos, lo cual corresponde a la suma del número de verdaderos positivos y falsos positivos. Por su parte, la exhaustividad se define como el número de verdaderos positivos sobre el total de elementos positivos, cantidad que puede calcularse como la suma del número de verdaderos positivos con falsos negativos. De esta forma, la precisión permite cuantificar el porcentaje de las predicciones positivas que fueron acertadas, mientras que la exhaustividad cuantifica el porcentaje de elementos positivos que fueron identificados correctamente. En el caso del ejemplo anterior, estas métricas habrían permitido dictaminar el reducido desempeño de la clasificación realizada por el algoritmo ya que, por no predecirse

ningún elemento como positivo, no habrían verdaderos positivos y por ende, tanto la precisión como la exhaustividad habrían sido de cero.

Por otra parte, las métricas para clases sesgadas presentan también la ventaja de que permiten apreciar el tipo de clasificación realizada por el algoritmo: Si se tiene una alta precisión, esto quiere decir que el algoritmo realiza predicciones confiables y certeras, pero que podrían no incluir a todos los sujetos de la población positiva, mientras que si se tiene una alta exhaustividad, la predicción podría no ser confiable para todos los casos, pero dentro de las predicciones realizadas como positivas, se encontrará un gran porcentaje de la población positiva. Ya que varios de los algoritmos de aprendizaje automático producen a su salida un número que luego se compara con un valor umbral (el cual es generalmente cero, como, e.g., en el caso del FLDA ilustrado en la Figura 17), dicho umbral puede modificarse para aumentar la precisión o exhaustividad [61]; e.g., si consideramos a la clase que corresponde a un valor mayor que cero como positiva, aumentar el valor de comparación de 0 a 0.4, aumentará la precisión, mientras que reducir dicho número aumentará la exhaustividad.

Sin embargo, también existen situaciones en las cuales no se desea lograr únicamente una precisión o exhaustividad significativa, sino que se desea optimizar ambos valores para que sean lo más grandes posible. En dichas situaciones, debe de utilizarse una tercera métrica para clases sesgadas, la cual es conocida como el puntaje F_1. Aunque una primera propuesta intuitiva para una métrica que tiene en consideración a la precisión y exhaustividad sería el promedio de estas dos, debe de observarse que esta no exige que la clasificación realizada sea precisa y exhaustiva a la vez, sino que podría ocurrir que solo una de las dos tome un valor elevado. Por consiguiente, la métrica correcta es el puntaje F_1 [61], el cual se define como se muestra en la siguiente ecuación:

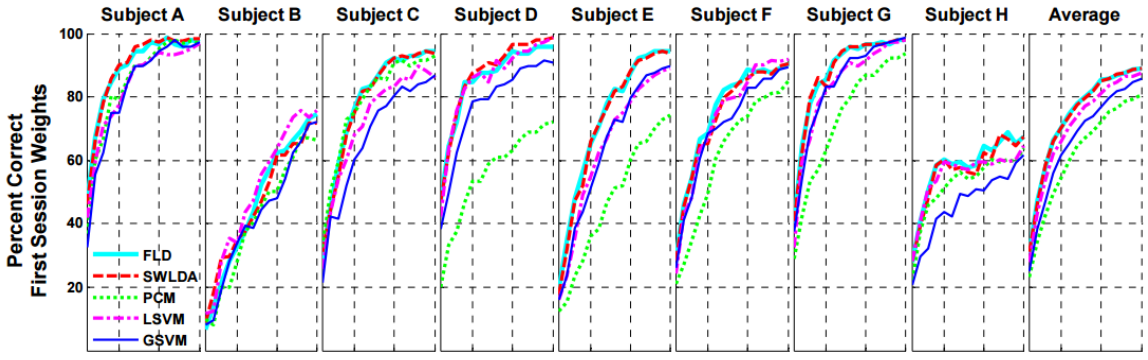
$$F_1 = \frac{2PR}{P + R}$$

Donde P es la precisión del algoritmo y R su exhaustividad. En dicha expresión, el producto de la precisión y la exhaustividad en el numerador hace que el puntaje F_1 no tome un valor cercano a 1 a no ser que ambos P y R sean cercanos a 1. De esta manera, esta métrica exige que el clasificador sea preciso y exhaustivo a la vez.

6. Desempeño de los algoritmos en la clasificación de la onda P300. Se han realizado diversos estudios en los cuales se ha comparado el desempeño de los algoritmos presentados anteriormente al momento de ser utilizados en la clasificación de señales de EEG para reconocer si estas contienen, o no, a la onda P300. Krusienski [44] realizó un estudio comparativo en el cual analizó diferentes algoritmos no solo por su desempeño en el número de clasificaciones correctas realizadas (ilustradas en la Figura 21), sino también por sus costos computacionales. Los autores discutieron que FLDA era un algoritmo sencillo, lo cual

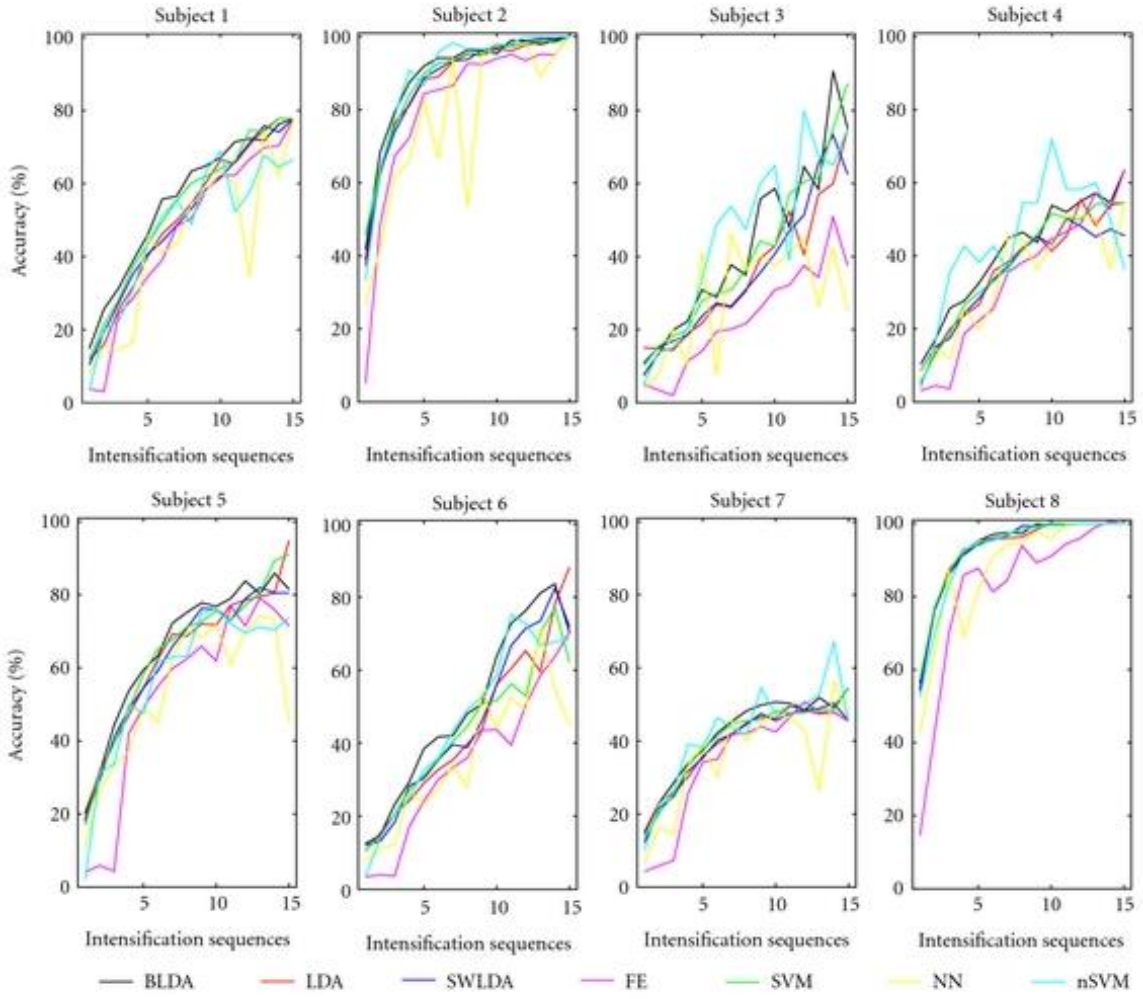
permitía un rápido entrenamiento e implementación del mismo; SWLDA presentaba la ventaja de seleccionar automáticamente a aquellas características de la señal que eran significativas para su correcta clasificación y los SVMs podían tener un rendimiento adecuado con pocos datos de entrenamiento, pero era complejo y lento de entrenar, además de que el SVM de *kernel* gaussiano era propenso a sobre-ajustar la información. De esta forma, los autores concluyeron que FLDA y SWLDA presentaban la mejor combinación de características de entrenamiento y desempeño para su implementación en la práctica, seguidos por el SVM [44].

Figura 21. Exactitud de cinco clasificadores para diferentes sujetos durante los experimentos de Krusienski [44]. En la figura, FLD corresponde a FLDA, LSVM a un SVM de *kernel* lineal y GSVM, a uno de *kernel* gaussiano, mientras que PCM corresponde a un clasificador basado en el coeficiente de correlación de Pearson.



Por otra parte, Manyakov [48], realizó un estudio sobre el desempeño de los algoritmos de aprendizaje automático al aplicarlos sobre mediciones de la onda P300 grabados de las señales EEG de personas que sufrían de diferentes discapacidades, las cuales incluían a sujetos que sufrían de esclerosis lateral amiotrófica, lesión de la arteria cerebral media y hemorragia subaracnoidea. Las exactitudes de los diferentes algoritmos obtenidas en este trabajo se ilustran en la Figura 22. La conclusión de dicho trabajo fue que el mejor clasificador fue el BLDA (*Bayesian Linear Discriminant Analysis*, un algoritmo similar a FLDA y SWLDA, pero que utiliza la regla de Bayes para crear un modelo equivalente a una versión penalizada de mínimos cuadrados ordinarios), seguido por el SVM de *kernel* gaussiano [48].

Figura 22. Exactitud de siete clasificadores para diferentes sujetos durante los experimentos de Manyakov [48]. En la figura, LDA corresponde a FLDA; SVM, a un SVM con *kernel* lineal; nSVM, a un SVM con *kernel* gaussiano; NN a una red neural *multilayer perceptron*, y FE y BLDA a dos métodos que no son tratados en este trabajo.



A pesar de los múltiples sistemas que se han propuesto para lograr una clasificación óptima de la onda de P300, ninguno de ellos ha resultado ser el algoritmo o método de clasificación definitivo [11]. Solo con los dos ejemplos dados de estudios realizados, se puede observar como las conclusiones difieren de un autor a otro: Mientras que Krusienski y su equipo no consideraron al BLDA y describieron al SVM de *kernel* gaussiano como un algoritmo con bajo desempeño que se sobre-ajustaba a los datos de entrenamiento, colocándose por debajo del FLDA, Manyakov y su grupo sí consideraron al BLDA y nombraron al SVM de *kernel* gaussiano como el segundo mejor clasificador, inclusive por encima del FLDA. Si bien estos estudios fueron realizados sobre diferentes poblaciones (en el primero se utilizaron personas sin discapacidades o enfermedades mentales, mientras que en el segundo sí se trabajó con personas con dichos padecimientos) lo cual justifica las variaciones entre los resultados, es un hecho que, como indica Cashero [11]:

> «Es difícil comparar directamente los diferentes métodos presentados en la literatura puesto que utilizan datos tomados de diferentes laboratorios y sujetos, los cuales pueden variar significativamente.»

Prueba de lo anterior se halla en la Figura 21 y Figura 22, donde se puede apreciar que, mientras algunos sujetos logran clasificaciones que llegan hasta casi el 100%, otros llegan hasta el 50 o 70%. Esto deja entrever que el desempeño del sistema no estará únicamente definido por el algoritmo de clasificación, sino también por el sujeto del cual se obtienen las señales, así como del equipo que se utiliza para realizar dichas mediciones.

E. INTERFACES CEREBRO COMPUTADORA

Las interfaces cerebro computadora (BCI, por las siglas en inglés de *Brain-Computer Interface*) son sistemas de comunicación que permiten al usuario transmitir comandos únicamente utilizando señales provenientes de su cerebro, sin requerir ningún tipo de actividad motora [44]. Para poder implementarlo, se requieren de dos componentes: Una señal cerebral que el sujeto sea capaz de controlar de manera confiable y sin realizar ningún movimiento muscular, así como clasificadores que puedan detectar dicha señal específica [11].

De acuerdo con Lotte [46], dentro de las señales cerebrales que pueden ser controladas por el usuario, se han propuesto varias alternativas, tales como la intención de movimiento del usuario, la imaginación de movimiento (e.g., imaginar que se mueve la mano derecha o el pie izquierdo), la realización de ciertas actividades mentales (e.g., realizar operaciones matemáticas o concentrarse en la escritura de una carta), las ondas cerebrales generadas por la relajación del usuario o la onda P300. Por otra parte, los clasificadores que se utilizan para detectar a las señales corresponden a los algoritmos de aprendizaje automático anteriormente presentados.

Sin embargo, las señales de EEG utilizadas en las interfaces cerebro computadora exhiben múltiples características que dificultan su clasificación [46]. Una de ellas es su reducida relación de señal a ruido, la cual exige el uso de técnicas de procesamiento de señales para mejorar la calidad de la señal de interés. En este trabajo, llamaremos cadena de procesamiento a una secuencia de técnicas de procesamiento de señales que son aplicadas sobre las señales de EEG disponibles para mejorar su relación de señal a ruido, con el objetivo de facilitar la tarea de clasificación al algoritmo de aprendizaje automático. Dos cadenas de procesamiento pueden presentar las mismas etapas de procesamiento de las señales, pero si dichas etapas son aplicadas en diferentes órdenes, se dice que las cadenas de procesamiento no son iguales.

Sin embargo, aún con la aplicación de cadenas de procesamiento sobre las señales de EEG, no resulta extraña la existencia de elementos que pertenecen a una clase pero que, debido a la presencia de ruido,

terminan pareciéndose más a elementos de la otra clase. Otro punto que dificulta la clasificación es que las señales de EEG son no estacionarias: i.e., sus características varían a través del tiempo, especialmente entre sesiones en las cuales las posiciones de los electrodos pueden variar [46]. Una tercera propiedad que causa problemas para la clasificación es la alta dimensión de los vectores de características usados para describir a las señales de EEG, lo cual está asociado con la maldición de la dimensión: mientras más características se tengan, también se deben de tener una mayor cantidad de datos de entrenamiento para poder obtener un clasificador adecuado. Sin embargo, las señales utilizadas en interfaces cerebro computadora también se caracterizan por contar con conjuntos de entrenamiento pequeños, debido a que la toma de dichos datos es un proceso que requiere de mucho tiempo y que es demandante para los sujetos [46]. También debe de considerarse la existencia de un fenómeno conocido como analfabetismo en las interfaces cerebro computadora, el cual es un término utilizado para describir el hecho de que las interfaces cerebro computadora no funcionan para un porcentaje significativo de la población (entre el 15% y 30%) [7]. Todos estos aspectos hacen que la tarea de clasificación de las señales de EEG sea una muy compleja que inclusive puede llegar a ser infructuosa para algunos usuarios.

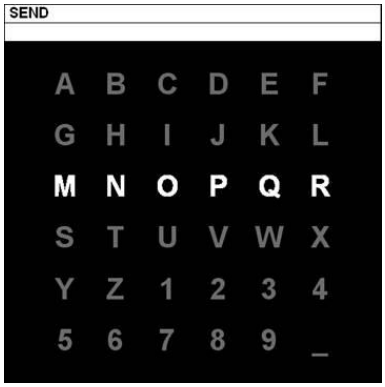
De acuerdo con Graimann, Allison y Pfurstscheller [34], las interfaces cerebro computadora pueden dividirse en dos tipos, síncronas y asíncronas, dependiendo de los márgenes de tiempo en los cuales el usuario puede utilizarlo. Las interfaces síncronas corresponden a aquellas que llevan una secuencia definida y que únicamente analizan las señales del usuario en ciertos momentos específicos: e.g., la interfaz puede mostrar al usuario que está próxima a iniciar la grabación de señales a través de un mensaje de pantalla o un sonido y grabar únicamente los 10 s posteriores a dicha marca de inicio. Este tipo de interfaces se utilizan generalmente para detectar señales cerebrales correspondientes a potenciales relacionados a eventos, ya que la interfaz estará mostrando un conjunto de estímulos y analizando las respuestas del usuario a estos. Por su parte, las interfaces asíncronas se encuentran analizando constantemente las señales cerebrales, dado que su objetivo es que el usuario utilice la interfaz en cualquier momento [34]. Este tipo de interfaces pueden ser utilizadas cuando las señales a analizar no requieren la presentación de estímulos, como en el caso de las oscilaciones neuronales, la realización de ciertas tareas mentales o la imaginación de movimiento.

Las interfaces cerebro computadora han gozado de un gran desarrollo e interés en las últimas décadas, siendo una de las principales causas de ello el hecho de que constituyen una oportunidad para mejorar la calidad de vida de personas que sufren de distintas discapacidades motoras. De tal cuenta que, en los últimos años, se han desarrollado sistemas que muestran potencial para controlar dispositivos externos, tales como sillas de ruedas, prótesis, computadoras o inclusive, aplicaciones dentro de casas inteligentes [11]. Un ejemplo notable de estas interfaces cerebro computadora es el teclado P300 [48], el cual se presentará a continuación.

1. Teclado P300. Es una interfaz cerebro computadora que fue propuesta inicialmente por Farwell y Donchin en 1988, permitiendo al usuario escribir una letra a la vez, utilizando únicamente sus señales

cerebrales [48]. Tal y como su nombre lo indica, la señal cerebral utilizada para identificar la letra deseada por el usuario es la onda P300, la cual se provoca de la siguiente manera: Se presenta al usuario una matriz con 36 elementos, organizados en seis columnas y seis filas, tal y como se muestra en la Figura 23. Cada una de estas filas y columnas son iluminadas una a la vez, a una frecuencia constante y de forma aleatoria, siguiendo una distribución de probabilidad uniforme. El usuario debe de concentrarse en la letra que desea escribir, siendo el evento de interés el que se ilumine el estímulo objetivo. Dicho estímulo objetivo puede ocurrir dos veces (una cuando se ilumine la fila donde se encuentra la letra deseada y otra, cuando se ilumine la columna) dentro de cada doce iluminaciones, por lo que tiene una probabilidad de ocurrir de 1 cada 6, lo cual ya se considera como una probabilidad baja. No está de más aclarar que las iluminaciones de las demás filas y columnas que no contienen a la letra de interés constituyen a los estímulos no objetivos.

Figura 23. Matriz mostrada al usuario en un teclado P300 [11].



De esta forma, cada vez que se ilumine la letra deseada, aparecerá una onda P300 entre las señales cerebrales del usuario, debido a que ha ocurrido un evento, impredecible y de baja probabilidad, que le interesaba al usuario [44]. Dada la baja relación de señal a ruido de la onda P300, se deben de realizar varias iluminaciones de todas las filas y columnas para obtener una respuesta promedio para cada una de ellas. Una vez se cuenta con dichas respuestas promedio, solo dos de ellas deberían de contener una onda P300: Una de ellas correspondiente a una fila y otra, a una columna. Conociendo la fila y columna para las cuales el usuario exhibió una onda P300, se puede determinar la letra en la cual él estaba concentrado [11].

No obstante, para que el sistema sea capaz de detectar la onda P300 del sujeto, es necesario que dicho sistema cuente con un clasificador que le permita identificar dicha onda. Dicho clasificador requiere de señales de EEG de ejemplo del usuario en cuestión, en las cuales se puedan apreciar las características de la onda P300 [11]. Para obtener las señales que servirán para la creación del modelo de clasificación, se realiza primero una serie de sesiones de entrenamiento. En ellas, se sigue el procedimiento descrito anteriormente, pero con la diferencia de que el usuario no elige las letras que escribirá, sino que estas son definidas previamente, de tal forma que la interfaz cerebro computadora sabrá cuál columna y fila debería de contener una onda P300. De esta manera, la interfaz asumirá que, efectivamente, dicha onda se encuentra en las señales

de EEG correspondientes a la fila y columna de la letra previamente acordada y extraerá señales de ejemplo, tanto para casos que corresponden al estímulo objetivo como a estímulos no objetivo, y utilizará dichas señales para entrenar el clasificador. Una vez entrenado el sistema, el usuario podrá elegir sus propias letras y estas deberían de ser identificadas correctamente por el clasificador [44]. De acuerdo con Cashero [11], existen estudios que indican que la onda P300 de un sujeto se mantiene estable durante un período de 40 semanas. Esto indica que, para un sujeto, las sesiones de entrenamiento recién descritas deben de ser realizadas al menos cada 40 semanas. Por otra parte, existen dos formas de realizar las predicciones una vez el clasificador se encuentra ya entrenado: Si dichas predicciones son todas realizadas luego de finalizada la medición de datos, se dice que la interfaz cerebro computadora funciona fuera de línea y que las predicciones correspondientes también fueron realizadas fuera de línea. Pero, si cada predicción es hecha al finalizar el conjunto de estímulos correspondientes a una misma letra, se dice que las predicciones son realizadas en tiempo real o, de manera equivalente, en línea.

Existen dos métricas de interés relacionadas con el desempeño de un teclado P300. La primera de ellas es la exactitud con la cual predice las letras indicadas por el usuario. Una de las problemáticas con dicha métrica es que, como se ha presentado anteriormente, depende en gran parte del usuario analizado. La segunda métrica de interés es la tasa de transferencia de información del sistema, a la cual definiremos como el número de palabras, por minuto, que pueden ser escritas por el usuario. Esta segunda métrica depende de varios parámetros relacionados con la estimulación del usuario. Uno de estos parámetros es el tiempo que dura la iluminación de cada una de las filas y columnas de la matriz. El otro parámetro es conocido como intervalo inter-estimulo (ISI, por las siglas en inglés de *Inter-Stimulus Interval*), el cual corresponde al tiempo que ocurre entre el final de una iluminación y el inicio de la siguiente [68]. La suma de estos dos parámetros da como resultado el tiempo ocurrido entre el inicio de dos iluminaciones, lo cual corresponde al período de cada estímulo. El inverso multiplicativo de dicho número indica la frecuencia de la estimulación.

Lo anterior implica que, para poder iluminar las 12 filas o columnas de la matriz, se requiere un tiempo de doce veces el período del estímulo. Además, como ya se mencionó con anterioridad, deben de realizarse múltiples iluminaciones de la matriz para poder obtener una respuesta promedio para cada una de las filas y columnas. En la literatura se halló que, generalmente, se utilizan 15 repeticiones durante la estimulación, como se puede observar en los estudios de Manyakov [48], Krusienski [44] y Cashero [11]. Esto quiere decir que, para poder predecir una letra, se debe de estimular al paciente durante 180 veces el período de estímulo y luego se debe de esperar a que el clasificador termine de procesar las señales para ofrecer una salida. Asumiendo un período de estimulación de 175 ms, se obtiene que se requieren de, al menos, 31.5 s para poder predecir una letra. Existen estudios en los cuales se ha determinado que la amplitud de la onda P300 incrementa mientras mayor sea la separación entre dos estímulos objetivo, la cual únicamente puede incrementarse si se aumenta el período de estimulación [32]. Por lo tanto, el parámetro de período de estimulación ofrece una alternativa entre hacerlo más largo para aumentar la amplitud de la onda P300, pero

incrementando el tiempo requerido para predecir una letra o, en contraposición, disminuirlo para reducir el tiempo requerido para la predicción, así como la amplitud de la onda P300.

Por otra parte, en la presentación de estímulos de un teclado P300, ocurren varios fenómenos que dificultan la correcta clasificación de las señales cerebrales [27]. Uno de estos fenómenos es el efecto de amontonamiento, en el cual diferentes estímulos (en este caso, la iluminación de cada una de las filas y columnas de la matriz) ocurren en un área muy pequeña, dificultando al usuario la diferenciación de los distintos estímulos [27]. Otro fenómeno es el problema de adyacencia, en el cual el usuario está centrando su atención en una de las letras pero la iluminación de filas y columnas vecinas captan su atención, generando una onda P300 que se registrará no para la letra de interés, sino para alguna cercana [27]. Las dos problemáticas recién presentadas pueden combatirse utilizando una menor cantidad de elementos en la matriz o usando una pantalla más grande para mostrar la matriz, con el fin de aumentar la separación entre los elementos [27]. Una tercera problemática surge del hecho de que, luego de varias escrituras en pantalla, el usuario se empieza a cansar, por lo que se le dificulta mantener su concentración en la letra. Para combatir esto, Fazel-Rezai y Ahmad [27], proponen:

> «[…] innovar en el diseño de los paradigmas visuales y hacerlo más sencillo para los usuarios. Otra forma de evitar la fatiga podría ser reducir el tiempo de escritura de cada letra […]»

Con el objetivo de combatir las dificultades presentadas, se han realizado diferentes modificaciones a la forma en la cual Farwell y Donchin presentaron los estímulos del teclado P300. Dicha forma de organizar e iluminar la matriz ha pasado a conocerse como el paradigma de fila y columna [27]. Dos de las variaciones realizadas a dicho paradigma se conocen como el paradigma de un solo carácter, en el cual se ilumina únicamente una letra a la vez dentro de la matriz de 6 x 6, y el paradigma de tablero de ajedrez, en el cual los elementos son iluminados por diagonales, con el objetivo de reducir el problema de adyacencia [27]. Otro paradigma alternativo es el de regiones, en el cual el usuario seleccionará primero una región y luego, dicha región contendrá diferentes letras para que se seleccione una de ellas. La ventaja que presenta este paradigma es que se tienen menos elementos en pantalla de forma simultánea, reduciendo el efecto de amontonamiento y el problema de adyacencia [27]. Un ejemplo de este paradigma se ilustra en la Figura 24.

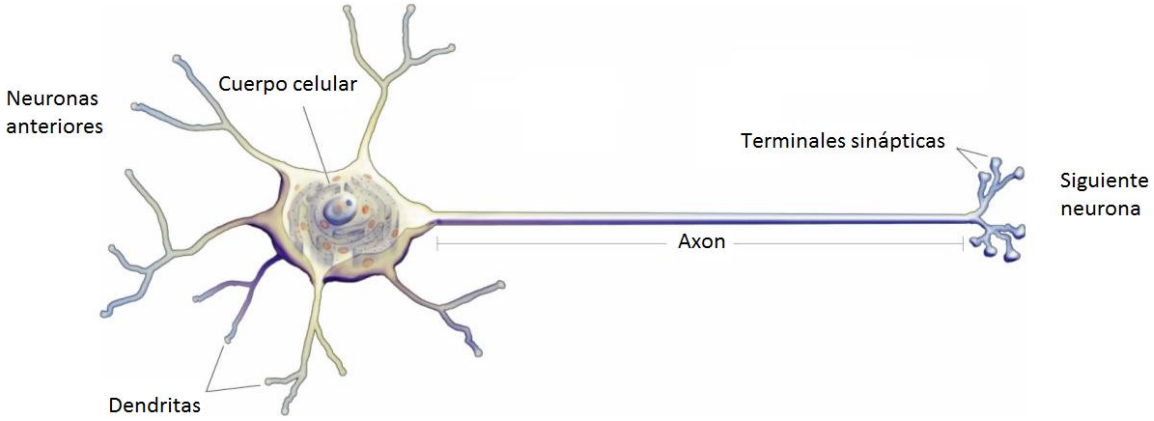
Figura 24. Pantalla presentada al paciente en un teclado P300 con el paradigma de regiones [27].

Cashero [11] indica que, a pesar de que el teclado P300 es uno de las interfaces cerebro computadora más estudiadas, aún se requiere de más trabajo para mejorar la velocidad y exactitud de este sistema, antes de que pueda ser implementado en la práctica.

F. ELECTROMIOGRAFÍA

1. Sistema nervioso. El sistema nervioso es el encargado de tres funciones principales: la actividad motora, funcionamiento sensorial y la comunicación entre células. Esto lo hace mediante el envío y recepción de estímulos eléctricos a través del cuerpo. El sistema nervioso está compuesto por neuronas interconectadas, la cual se conoce como red neuronal. Estas son células que envían y reciben impulsos eléctricos a través de ellas para ejecutar las tres funciones principales [25].

Figura 25. Estructura de una neurona [16].



Entre las partes interior e exterior de las neuronas (dentro y fuera su membrana), existe una diferencia de potencial (debido a las concentraciones de sodio y potasio dentro y fuera de ella). Este potencial se conoce como potencial de membrana (membrane potential en inglés). Las neuronas modifican estas concentraciones para variar este potencial y así enviar impulsos eléctricos a través del axón y hacia las terminales sinápticas. Estos impulsos se conocen como potenciales de acción [85].

Como se mencionó anteriormente, estos son enviados para accionar la actividad motora, funcionamiento sensorial y comunicación entre células. Las potenciales de acción enviadas debido al accionamiento de la actividad motora se conocen como potencial de acción de las unidades motoras (MUAP) [99] o señales electromiográficas (EMG) y la medición de estas se conoce como electromiografía.

La amplitud de las señales mioeléctricas puede ser relativamente baja, incluso para un microcontrolador o una computadora, teniendo un rango entre 5 mV y 15 mV. Aunque puede variar según el músculo a evaluar.

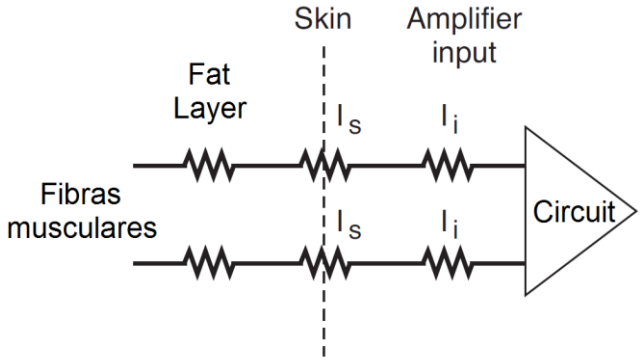
Para que estas puedan ser utilizada es necesario amplificarla, esto se realiza con amplificadores de instrumentación diseñados para tener ganancias relativamente grandes, entre 600 a 50,000 veces. Para reducir la impedancia de la señal y proveer una amplificación inicial es recomendable implementar una etapa de pre amplificación [62].

2. Electrodos para mediciones no invasivas. El uso de electrodos no invasivos proveen una manera fácil y seguro de obtener información acerca de cómo se están accionando los músculos. Sin embargo, es necesario usar varios electrodos para poder diferenciar entre las señales siendo enviadas a través de los músculos. Aunque es un método más seguro que el invasivo, este tiene la desventaja de que energía de otro grupo de músculos (distinto al grupo de interés) podría mezclarse en la señal y se tendrían mediciones incorrectas. Por esto es necesario posicionar bien los electrodos y escoger un tamaño adecuado (de tal manera que no intersecten varios grupos de músculos pero que sí abarque una buena porción del músculo de interés.) [18].

A la hora de escoger los electrodos a utilizar, se debe de tomar en cuenta lo siguiente: grasa o tejido adiposo entre los electrodos y el músculo de interés; impedancia de la piel; tamaño y material del electrodo; y, el cable que une el electrodo con el circuito captador. La grasa influye en que funciona como un aislante (aumenta la resistencia), por lo que entre más grasa hay entre el electrodo y el músculo, más disminuye la amplitud de la señal que llega al circuito. Lo mismo sucede con la piel y el cable, por lo que a veces se recomienda limpiar la piel con alcohol (disminuye la impedancia) y mantener los cables lo más corto posible [18].

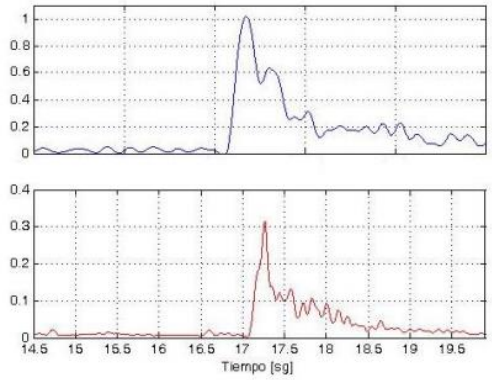
El tamaño del electrodo influye debido a que este varía la cantidad de fibras musculares que se medirán. Entre más pequeño el electrodo, más selectividad tendrá. Es decir, entre más pequeño, menos músculos tomará en cuenta (será más específico a una región de músculos). El material del que está hecho el electrodo también influye porque cambia la impedancia del mismo. Se desea que el electrodo capte las señales sin que este voltaje disminuya, por lo que la impedancia de este debe ser baja (ver Figura 26) [18].

Figura 26. Impedancias presentes en la captación de señales EMG [18].



3. Patrones EMG trifásicos. Al momento de clasificar la actividad motriz existen patrones EMG trifásicos característicos, que involucran los músculos flexores y extensores. Los cuales consisten en una sucesión de músculos: agonista, antagonista y agonista del movimiento, donde el primer movimiento causa un sobrepaso de la señal emg, al igual que el segundo antes de estabilizarse. Estos patrones se presentan principalmente en los movimientos rápidos [79].

Figura 27. Activación trifásica del bíceps y tríceps, la imagen superior muestra la activación del bíceps y la inferior la activación del tríceps [79].



4. Configuración de electrodos superficiales. Las mediciones se clasifican por la configuración de los electrodos. En la primera configuración, la monopolar, se utilizan dos electrodos: el de referencia que es colocado lejos del lugar de medición y el electrodo de medición. En esta se mide la diferencia entre ambos electrodos para obtener la señal; sin embargo no es recomendable utilizarla, ya que es más susceptible al ruido, a pesar de su simplicidad, ver Figura 28. En la segunda configuración, la bipolar se utilizan dos electrodos para realizar la medición, separados cerca de 2 cm con la misma orientación de las fibras musculares, más un electrodo de referencia. Al medir la diferencia entre ambos electrodos de medición se reduce el ruido, por lo que es más común implementarla, ver Figura 29. Por último, se encuentra la

configuración multipolar, en la cual se utilizan más de dos electrodos de medición y más de dos etapas de amplificación diferencial [99].

Figura 28. Configuración monopolar de los electrodos [99].

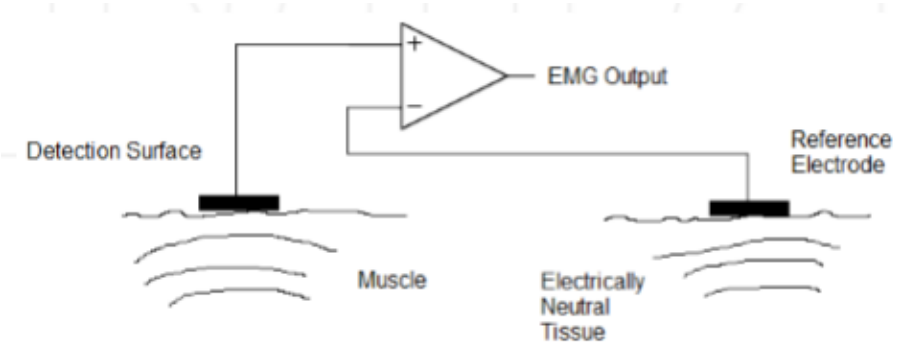
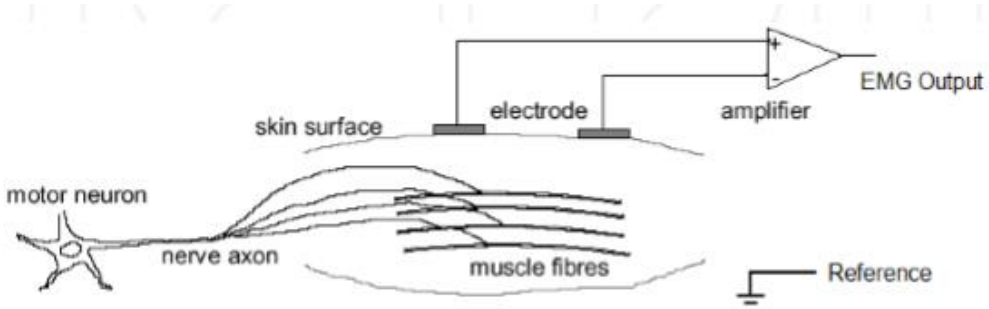


Figura 29. Configuración bipolar de los electrodos [99].

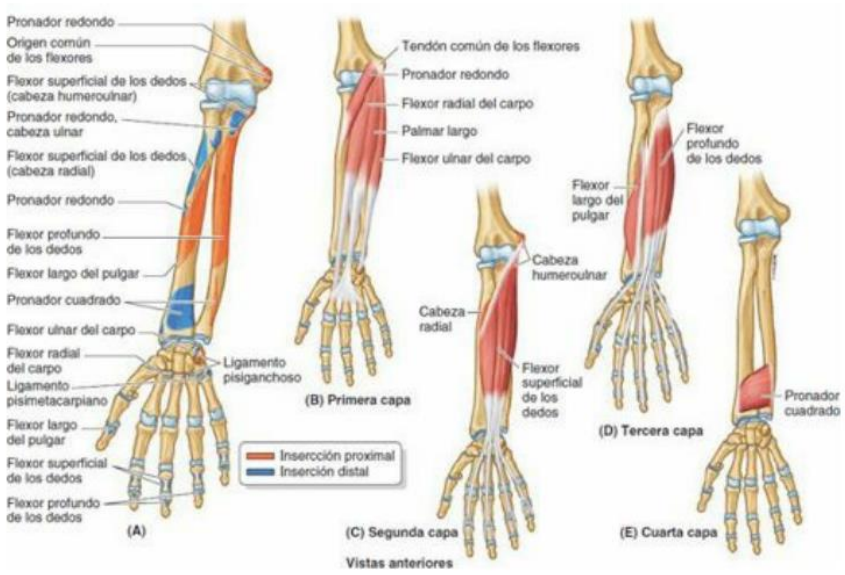


G. MÚSCULOS DEL ANTEBRAZO

Los músculos del antebrazo se separan en los flexores y los extensores, los cuales realizan movimientos opuestos. Los flexores se encuentran ubicados en el compartimiento anterior del antebrazo y tienen aproximadamente el doble de masa y fuerza que los músculos extensores, mientras que los extensores se encuentran ubicados en el compartimiento posterior del mismo. Los tendones de los músculos mantienen su posición por medio de ligamentos (Retináculo flexor y extensor), ver Figura 30 [54].

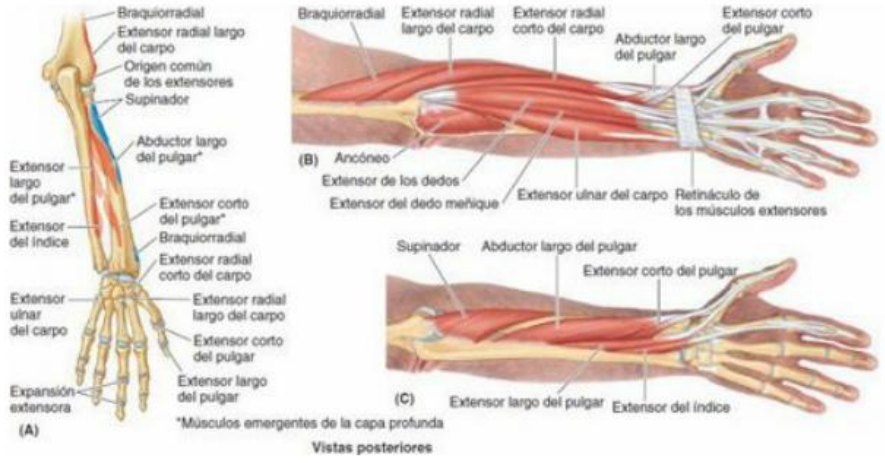
1. Músculos flexores del antebrazo. Los músculos flexores se dividen en tres capas: la capa superficial, donde se encuentran los músculos relacionados al movimiento de la muñeca; la capa intermedia, donde se encuentra el flexor superficial de los dedos; y la capa profunda, donde se encuentra el flexor profundo de los dedos y el flexor largo del pulgar. El musculo flexor profundo se encarga de los movimientos lentos, mientras que el flexor superficial de los movimientos rápidos [54].

Figura 30. Capas de los músculos flexores del antebrazo [54].



2. Músculos extensores del antebrazo. Los músculos extensores tambien se dividen en tres grupos: Los músculos que extienden o abducen la muñeca; los músculos que extienden los cuatro dedos mediales; y los que extienden o abducen el pulgar. En este caso solo existen dos capas, la superficial y la profunda, ver Figura 31 [54].

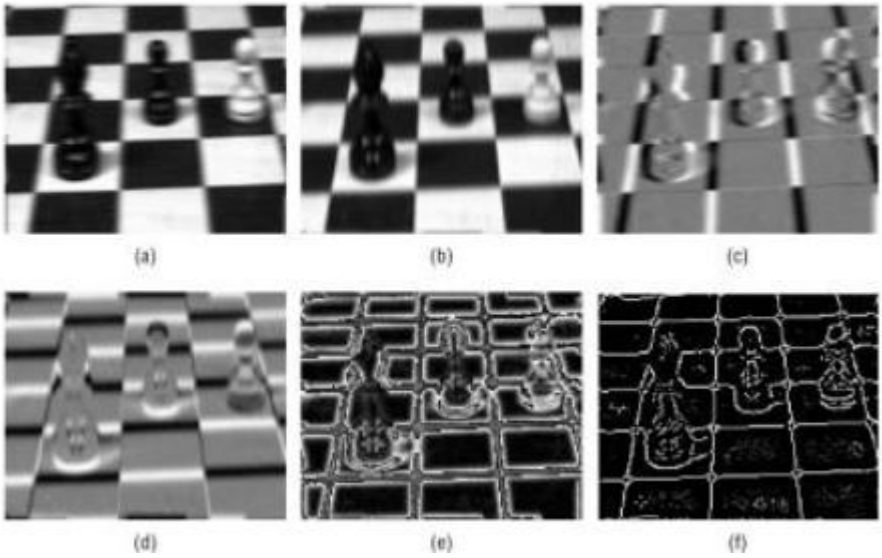
Figura 31. Capas de los músculos extensores del antebrazo [54].



H. VISIÓN POR COMPUTADORA

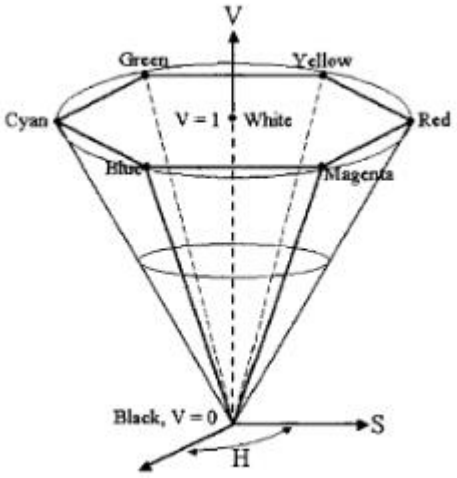
La visión por computadora es la percepción que tiene una computadora sobre un entorno tridimensional. Este se obtiene analizando forma, tamaño y color de los objetos a su alrededor, por medio de imágenes. Con esta se puede detectar rostros, objetos, bordes y colores, entre otras cosas y crear una aproximación de las imágenes en tres dimensiones [86]. Las computadoras procesan las imágenes por medio de matrices, en las cuales cada valor asignado representa una componente del color o matiz. Aunque ambos se pueden describir por medio de nombres como rojo, azul, violeta, amarillo, etc. El blanco, gris y negro también son considerados como colores, pero no como matices. En pocas palabras, existen dos clases de colores, los colores cromáticos (matices) y los colores acromáticos (e.g. blanco, negro y escalas de grises). Por medio de los cambios bruscos de colores, las computadoras pueden interpretar ejes o contornos, los cuales definen los objetos, ver Figura 32 [93].

Figura 32. Imagen de un tablero de ajedrez, se muestran los resultados de la detección de ejes por medio de una convolución Gaussiana [93].



Por ejemplo, en una imagen RGB, se utilizan tres valores por pixel, rojo, verde y azul respectivamente. Estos indican la saturación da cada uno de estos colores. También existen otros espacios de color como el HSV o HSI, donde se utilizan parámetros: H, matiz; S, saturación; y V o I que indica el valor o intensidad. En la Figura 33, se muestra el espacio de color HSV, que está definido como un espacio cónico donde el ángulo indica la matiz, la componente radial la saturación y la componente vertical el valor [93].

Figura 33. Representación del espacio de color HSV [93].



I. CUADCÓPTERO AR DRONE 2.0

1. Especificaciones generales de AR Drone 2.0

a. Cámara. El cuadcóptero (o drone) cuenta con una cama HD de 720 pixeles (graba a 30 FPS) que puede transmitir el video en tiempo real, tomar fotos, etc. La transmisión es de baja latencia y el lente de la cámara tiene un rango de visión de hasta 92º. Este puede transmitir el video a través de una conexión inalámbrica (Wi-Fi) o en una memoria USB [67].

b. Estructura robusta. El drone cuenta con una carcasa para exteriores (ver Figura 34) y una carcasa para interiores (ver Figura 35). El peso total del drone con la carcasa para exteriores es de 380 gramos. El peso total del drone, agregando la carcasa para interiores es de 420 gramos. Cuenta con tubos de fibra de carbono que sostienen los motores de las hélices (4 en total). Cuenta con una espuma entre los tubos y el centro inercial (de esta manera lo aísla de vibraciones de los motores) [67].

Figura 34. Carcasa para exteriores.

Figura 35. Carcasa para interiores.

Figura 36. AR Drone 2.0 configurado para volar adentro.

c. Electrónica y sistema operativo. Trabaja con el sistema operativo Linux 2.6.32. Este cuenta con: procesador de 1 GHz; RAM de 1GB que trabaja a 20MHz; puerto USB 2.0; conexión Wi-Fi; giroscopio de 3 ejes con una precisión de 2000º/segundo; acelerómetro de 3 ejes con una precisión de 50 mg; magnetómetro

de 3 ejes con una precisión de 6º; sensor de presión con una precisión de 10 Pascales; sensores ultrasónicos para medir la altitud; y, una cámara vertical para medir la velocidad de avance [67].

d. Motores. Cuenta con 4 motores sin escobillas de 14.5W y 28,500 rpm. Cada motor tiene: un cojinete de microbola; un engranaje de Nylatron (reductor de 8.75); un eje de transmisión de acero templado; un cojinete de bronce autolubricado; un CP AVR de 8 MIPS por controlador de motor; parada de emergencia; controlador de motor programable; controlador electrónico del motor resistente al agua [67].

e. Vuelo. El drone tiene cuatro rotores o hélices. En este tipo de sistema, es necesario tener cierta dirección de giro en las hélices para que este pueda volar bien. Esto básicamente consiste en que cada par opuesto de hélices gira en la misma dirección. Un par gira en el sentido de las agujas del reloj y par restante gira en sentido contrario a las agujas del reloj (ver Figura 37 y Figura 38).

Figura 37. Sentidos de giro necesarios para elevar el drone [66].

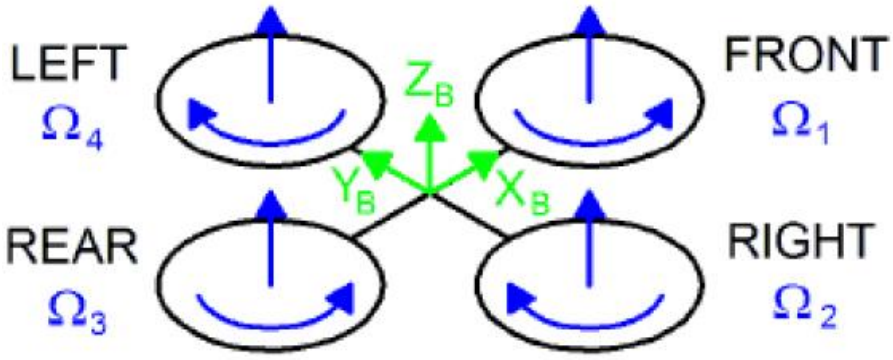
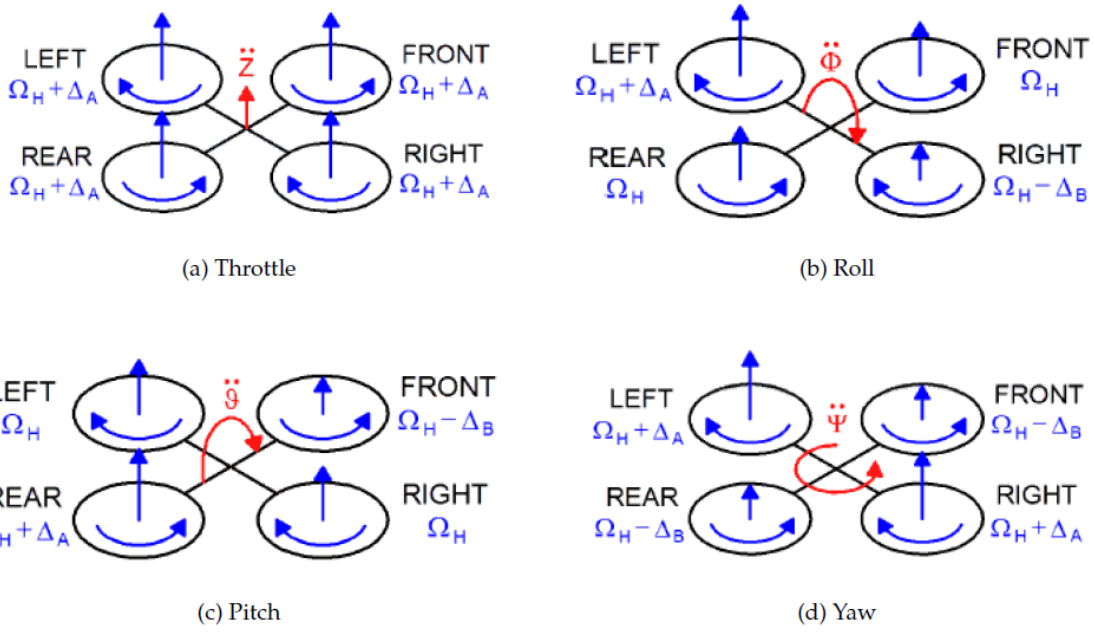


Figura 38. Distintas configuraciones en las hélices para crear ciertos movimientos [66].



El drone también cuenta con un sistema de parada de emergencia. Este funciona detectando si alguno de los motores o hélices encuentra algún obstáculo (choca contra algún objeto) o se encuentra bloqueado. Al detectar esto, apaga todos los motores y enciende una bandera de emergencia (el cual debe de ser reseteado para poder volar el drone nuevamente). Los sensores que trae el drone permiten leer datos como: altitud, velocidad de subida, rotación en roll, pitch, yaw, etc. El drone también trae controles o comandos predefinidos que sirven para despegar, aterrizar, etc. y programar otros tipos de movimientos [66].

2. Programación de AR Drone 2.0. Al encender el drone, este crea una red inalámbrica (por default este es ardrone2_xxx). Este crea una dirección IP (internet protocol) poco común y que usualmente no está en uso (192.168.1.1). Posteriormente, el usuario se debe conectar a esta red y enviar comandos a dicha dirección IP. El drone automáticamente le genera una dirección IP (192.168.1.x) para el usuario para establecer comunicación. De este momento en adelante, el usuario puede obtener información del drone y enviar comandos [66].

La comunicación entre el usuario y el drone ocurre sobre cuatro puertos, cada uno con una función específica. Esta comunicación se puede dar con UDP (User Datagram Protocol) o TCP (Transmission Control Protocol) y se termina la comunicación si pasan más de dos segundos sin enviar comandos al drone [66]:

- UDP 5556: Envía comandos al drone.
- TCP 5555: Transmisión del video.
- UDP 5554: Envia datos de navegación (altitud, pitch, nivel de batería, etc.).
- TCP 5559: Recibe datos de configuración del drone.

J. CINEMÁTICA DE BRAZOS ROBÓTICOS

1. Modelado de brazos robóticos. La cinemática es la rama de la mecánica que estudia el movimiento de un cuerpo o conjunto de cuerpos sin considerar su masa o las fuerzas que actúan sobre él. Un brazo robótico está compuesto por eslabones y articulaciones. Una articulación puede unir dos o más eslabones y permite que el eslabón siguiente se mueva con respecto al eslabón anterior. Al conjunto de eslabones y articulaciones se le conoce como una cadena. Usualmente la base de esta está fija y el último eslabón de la cadena sostiene el efector final o herramienta que se desea controlar [17].

Las articulaciones pueden ser de tipo cilíndricas o revolutas (rotación sobre un eje) o de tipo prismático (traslación en la dirección de un eje). El accionamiento de las articulaciones cambia la posición (angular o lineal) del siguiente eslabón respecto al eslabón anterior. El brazo robótico puede ser descrito por los tipos de articulaciones que lo componen R para rotacional y T para traslacional. Sin embargo, esta nomenclatura

solo describe el tipo de movimiento que tiene un eslabón con respecto al anterior, pero no describe traslaciones y rotaciones iniciales o fijas que tiene un eslabón con respecto al otro [17].

Existen los marcos de coordenadas, los cuales son una serie de ejes ortogonales entre sí (x,y,z) los cuales intersectan en algún punto (conocido como el origen). El marco de coordenadas ayuda a describir la posición y orientación de algún objeto respecto a ella. Una manera de describir a una cadena es colocar marcos en cada eslabón que indican la posición y orientación con respecto al marco anterior (eslabón anterior). De esta manera se podrá obtener la pose del efector final respecto al marco global. Es decir, se sabrá la posición y orientación del efector final con respecto al marco de la base de la cadena. Para obtener la pose del efector final con respecto al marco global, es necesario realizar una serie de transformaciones entre cada marco [17]:

$$^{A}P = {}^{A}T_{B}\,{}^{B}P \tag{8}$$

Equivalente a:

$$\begin{bmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{A}R_{B} & t \\ 0_{1X3} & 1 \end{bmatrix} \begin{bmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \\ 1 \end{bmatrix} \tag{9}$$

Donde:

^{A}P y ^{B}P representan las coordenadas x, y, z respecto al marco $\{A\}$ y $\{B\}$ respectivamente.

$^{A}T_{B}$ representa la pose relativa del marco $\{B\}$ respecto al marco $\{A\}$.

^{A}x, ^{A}y, ^{A}z representan las coordenadas x, y, z con respecto al marco $\{A\}$.

^{B}x, ^{B}y, ^{B}z representan las coordenadas x, y, z con respecto al marco $\{B\}$.

t es el vector de traslación entre los marcos de coordenadas.

$^{A}R_{B}$ es la matriz de rotación que representa la orientación entre los marcos de coordenadas.

Para obtener la matriz de rotación, es necesario analizar las rotaciones que han sido aplicadas al marco original para llegar al marco del efector final. Dependiendo de estas rotaciones, se tendrá que realizar las multiplicaciones de las siguientes matrices en el orden que fueron llevadas a cabo [17]. Ejemplo: si la rotación fue en Y, luego X y luego Z, entonces la matriz de rotación se calculará con $R_{y}(\theta_1)R_{x}(\theta_2)R_{z}(\theta_3)$.

$$R_{x}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{10}$$

$$R_{y}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{11}$$

$$R_{z}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

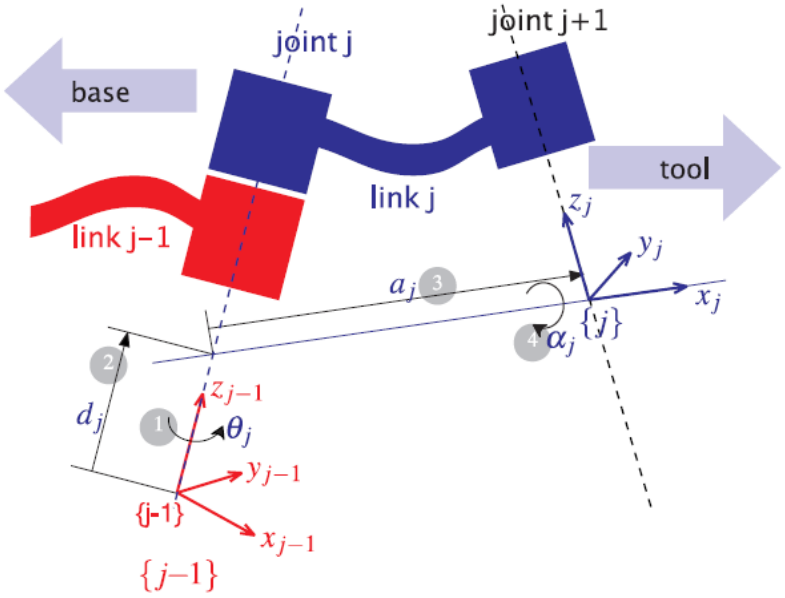
Donde:

R_x, R_y, R_z representan las rotaciones sobre los ejes x, y, z respectivamente.

θ representa el ángulo de rotación sobre ese eje.

2. Notación Denavit-Hartenberg. La notación Denavit-Hartenberg se introdujo en 1955 por Jacques Denavit y Richard Hartenberg. Es una manera sistemática de describir la geometría de una cadena y facilita la implementación de las transformaciones mencionadas anteriormente. Para utilizar esta notación, se tiene que realizar la siguiente suposición y análisis. Se tiene una cadena con N articulaciones (numeradas desde 1 hasta N), entonces existen N+1 eslabones (numeradas desde 0 hasta N). El eslabón 0 es la base de la cadena y el eslabón N contiene el efector final. La articulación j conecta el eslabón j-1 con el eslabón j, por lo que se puede decir que la articulación j mueve al eslabón j. Un eslabón puede ser especificado por dos parámetros: su largo a_j; y, su twist α_j, el cual define el ángulo de rotación del sobre el eje x. Una articulación también tiene dos parámetros: el offset d_j, el cual define la distancia de una articulación a otra sobre el eje de la articulación; y, el ángulo de rotación θ_j de un eslabón respecto al anterior sobre el eje de la articulación (eje z). Existe un quinto parámetro que indica el tipo de articulación (revoluta o prismática). Para poder aplicar esta notación es necesario cumplir con los siguientes requerimientos: el marco de coordenadas j está ubicada al final del eslabón j; y, el eje de la articulación j está alineada con el eje z [17].

Figura 39. Aplicación de notación Denavit-Hartenberg [17].



La transformación del eslabón j-1 al eslabón j está definido por las siguientes transformaciones:

$$^{j-1}A_j(\theta_j, d_j, a_j, \alpha_j) = T_{Rz}(\theta_j)T_z(d_j)T_x(a_j)T_{Rx}(\alpha_j) \quad (13)$$

Equivalente a:

$$^{j-1}A_j = \begin{bmatrix} \cos(\theta_j) & -\sin(\theta_j)\cos(\alpha_j) & \sin(\theta_j)\sin(\alpha_j) & a_j\cos(\theta_j) \\ \sin(\theta_j) & \cos(\theta_j)\cos(\alpha_j) & -\cos(\theta_j)\sin(\alpha_j) & \alpha_j\sin(\theta_j) \\ 0 & \sin(\alpha_j) & \cos(\alpha_j) & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

Debido a la notación de la ecuación (13), es necesario realizar las transformaciones en ese orden. Es decir, primero se trabaja la rotación θ_j, luego la traslación d_j, luego la traslación a_j y luego la rotación α_j.

Cuadro 2. Resumen de los parámetros Denavit-Hartenberg y su significado [17].

Parámetro	Símbolo	Significado físico
Ángulo de articulación	θ_j	Ángulo entre los ejes x_{j-1} y x_j sobre el eje z_{j-1}.
Offset del eslabón	d_j	Distancia del origen del marco $j-1$ al eje x_j sobre el eje z_{j-1}.
Longitud del eslabón	a_j	Distancia entre los ejes z_{j-1} y z_j sobre el eje x_j.
Giro del eslabón	α_j	Ángulo desde el eje z_{j-1} al eje z_j sobre el eje x_j.
Tipo de articulación	σ_j	$\sigma_j = 0$ para articulación de tipo revoluta y $\sigma_j = 1$ para articulación de tipo prismático.

$$^0T_N = {}^0A_1\,{}^1A_2 \ldots {}^{N-1}A_N \quad (15)$$

Donde:

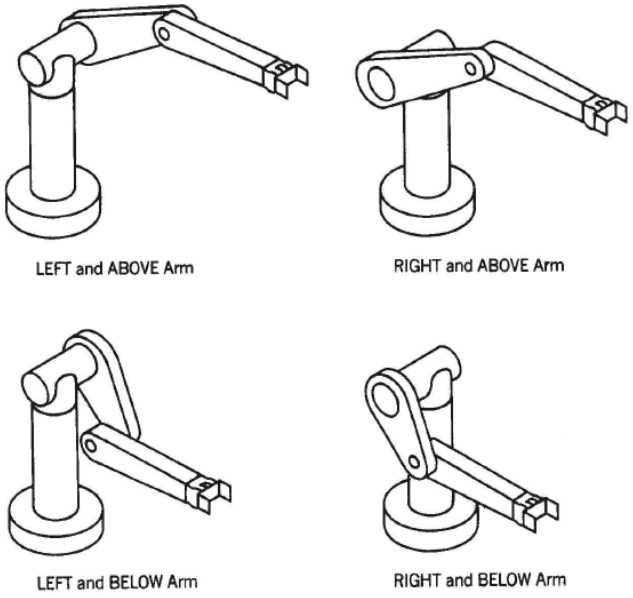
^0T_N representan las rotaciones sobre los ejes x, y, z respectivamente.

θ representa el ángulo que se rotó sobre ese eje.

3. Cinemática directa. Esta consiste en determinar la posición y orientación del efector final dado las posiciones de las articulaciones. Es decir, se determina la pose del efector final conociendo los ángulos y desplazamientos de las articulaciones [84].

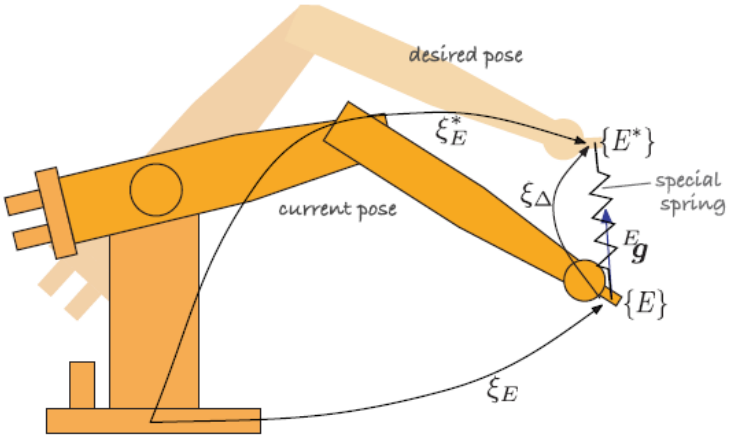
4. Cinemática inversa. Como lo indica el nombre, es el caso inverso a la cinemática directa. Consiste en determinar las posiciones de las articulaciones (ángulos y desplazamientos) dado las posiciones y orientaciones del efector final. Se puede dar el caso en el que varias configuraciones o posiciones de las articulaciones podrían cumplir con la pose del efector final. En estos casos, es necesario restringir el movimiento de algunas articulaciones para obtener un comportamiento deseado (ver Figura 40) [84].

Figura 40. Mismas soluciones con ángulos diferentes de los actuadores pero la misma pose [84].



5. Cinemática inversa con un método numérico. El Jacobiano es el equivalente matricial de una derivada. Con el Jacobiano uno puede relacionar las velocidades de las articulaciones con las velocidades traslacionales y angulares del efector final. De manera análoga, el Jacobiano puede relacionar fuerzas y torques (vector g) de las articulaciones con las fuerzas y torques del efector final (también conocido como "wrench"). El método numérico empleado utiliza el wrench actuando en el efector final para llevar el brazo a una posición deseada (ver Figura 41) [17].

Figura 41. Wrench aplicado para llegar a una pose deseada [17].



A continuación se encontrará la deducción matemática para utilizar el método numérico [17]. Basado en la Figura 41, se nota que el wrench es proporcional a la resta de las dos poses (deseada y actual), por lo que:

$$^{E}g = \gamma\Delta(\varepsilon_E, \varepsilon_E^*) \tag{16}$$

Donde:

γ es una constante de proporcionalidad.

ε_E es la pose actual, el cual se calcula con la cinemática directa de $q\langle k\rangle$.

$q\langle k\rangle$ es la estimación actual de la cinemática inversa.

$$Q\langle k\rangle = {}^{N}J(q\langle k\rangle)^{T}\,{}^{E}g\langle k\rangle \tag{17}$$

Donde:

$Q\langle k\rangle$ son las fuerzas y torques de articulación.

${}^{N}J(q\langle k\rangle)$ es el jacobiano de transformación.

$$\dot{q}\langle k\rangle = \frac{Q\langle k\rangle}{B} \tag{18}$$

Donde:

$\dot{q}\langle k\rangle$ son las velocidades de articulación.

B es el coeficiente de amortiguación.

$$q\langle k+1\rangle = \alpha\dot{q}\langle k\rangle + q\langle k\rangle \tag{19}$$

Donde:

α es una constante.

Este procedimiento se itera hasta la magnitud del wrench ${}^{E}g\langle k\rangle$ sea suficientemente pequeña. La ecuación (17) se modifica si se tiene un brazo sub-actuado. Se utiliza un vector máscara "m" para delimitar esta ecuación:

$$Q\langle k\rangle = {}^{N}J(q\langle k\rangle)^{T}\mathrm{diag}(m)\,{}^{E}g\langle k\rangle \tag{20}$$

Donde:

m es un vector de la forma $[t_x\ t_y\ t_z\ r_x\ r_y\ r_z]$ donde representan traslación y rotación en x, y, z.

K.ROBOT HUMANOIDE NAO

1. Especificaciones generales de NAO. NAO es un robot humanoide creado por Aldebaran Robotics en el año 2006. NAO actualmente tiene 25 grados de libertad (motores eléctricos). Tiene dos cámaras, cuatro micrófonos, sensores ultrasónicas, dos emisores y receptores infrarojos, etc. (ver Figura 42). Debido a su sistema operativo (NAOqi OS) y su código fuente, es posible controlar cada uno de sus motores. La programación de NAO se lleva a cabo ya sea por conexión inalámbrica o cable Ethernet [1].

Figura 42. Componentes de NAO [2].

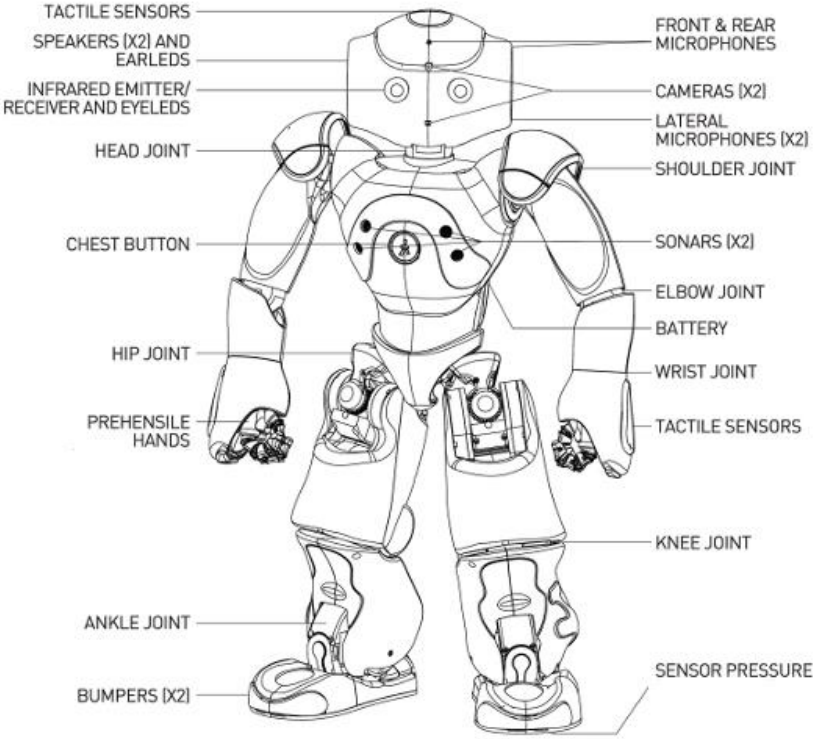
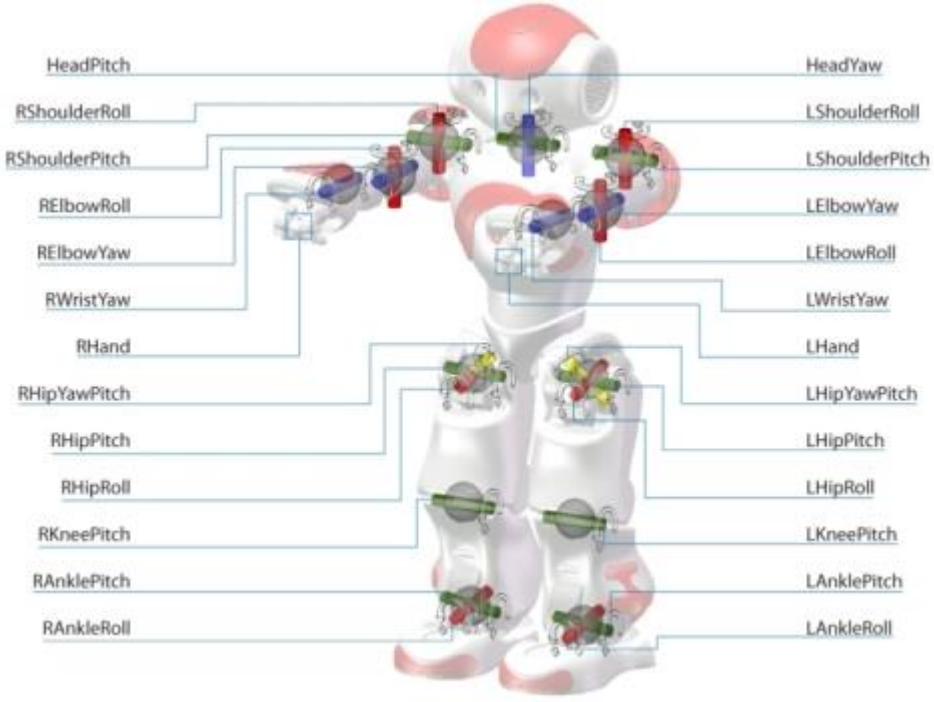


Figura 43. Actuadores y articulaciones presentes en NAO [2].



2. Especificaciones técnicas de NAO. Como se puede apreciar en la Figura 43, NAO se puede representar por medio de un una cadena, es decir, un conjunto de eslabones y articulaciones interconectadas. Por esto mismo se puede utilizar la teoría mencionada en la sección anterior (C. CINEMÁTICA DE BRAZOS ROBÓTICOS). Para simplificar el modelado de las cadenas, se puede utilizar la notación Denavit-Hartenberg. Sin embargo, NAO tiene su propio marco de referencia y convención de signos y dirección de giro. Es decir, 45º en la articulación LShoulderPitch indica un giro de 45º en cierta dirección con respecto a cierto eje. Por lo tanto, para obtener un movimiento deseado en NAO, es necesario seguir esta convención. A continuación se encuentran las especificaciones técnicas de NAO, estas se deben de respetar para obtener el movimiento deseado en NAO.

Figura 44. Dimensiones generales de NAO [2].

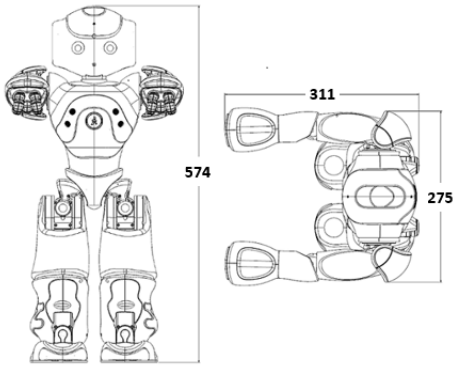


Figura 45. Convención de signos de NAO [2].

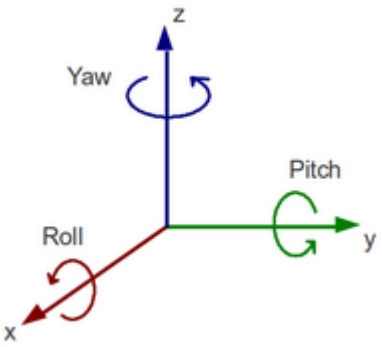


Figura 46. Marco de referencia de NAO [2].

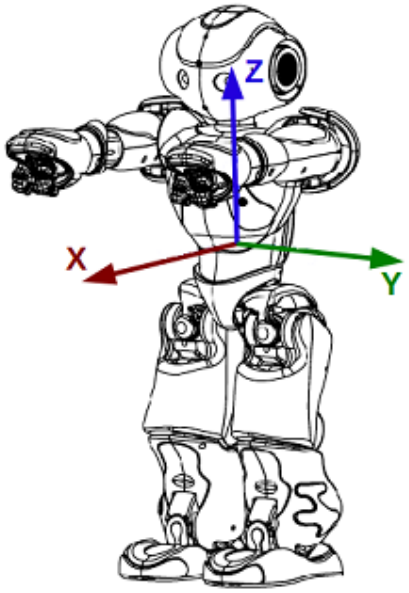


Figura 47. Dimensiones de la articulaciones principales de NAO [2].

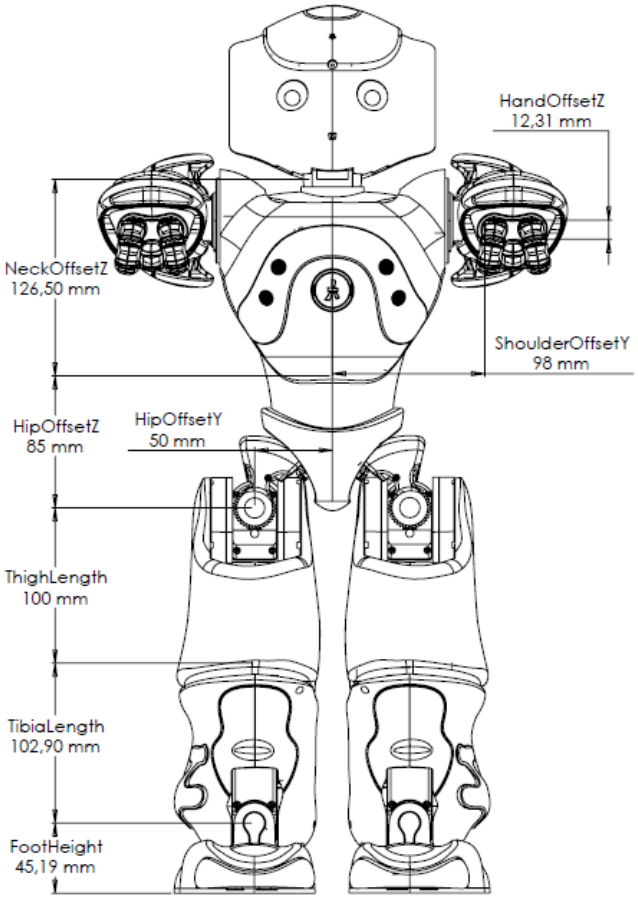


Figura 48. Vista superior de las dimensiones de las articulaciones principales de NAO [2].

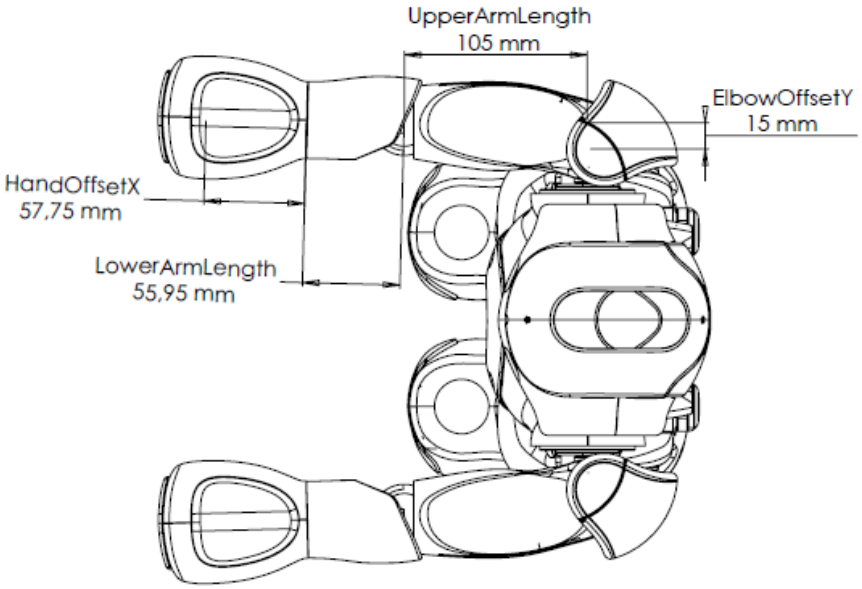


Figura 49. Límites y sentidos de giro de articulaciones en la cabeza de NAO [2].

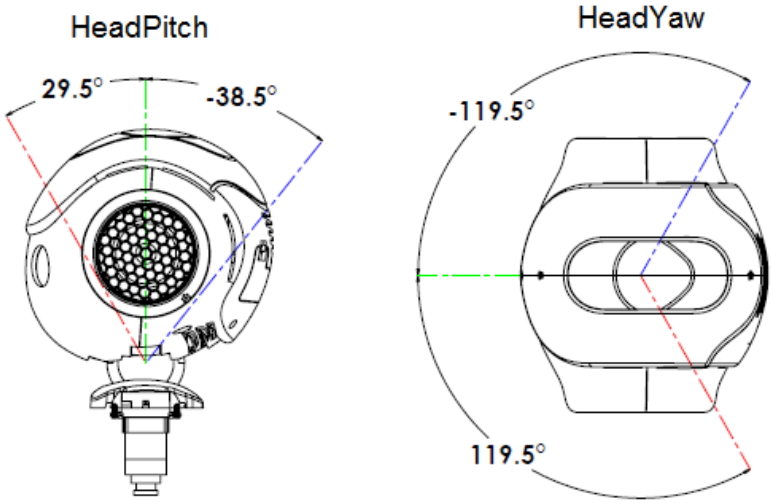


Figura 50. Límites y sentidos de giro de articulaciones en el brazo izquierdo de NAO [2].

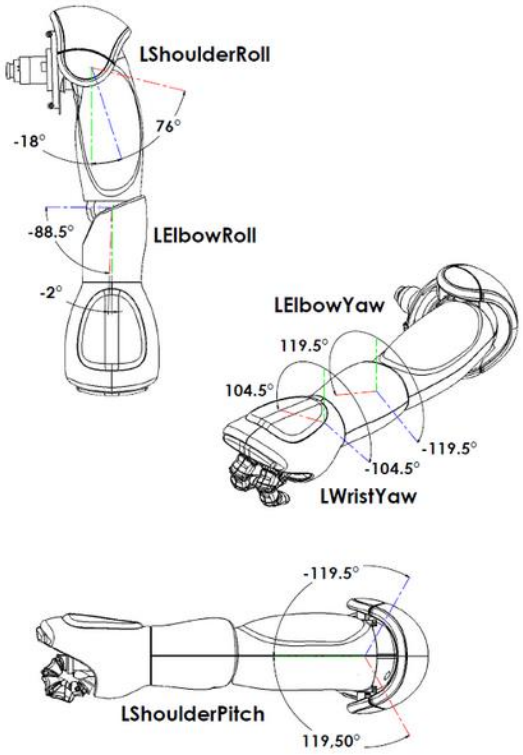
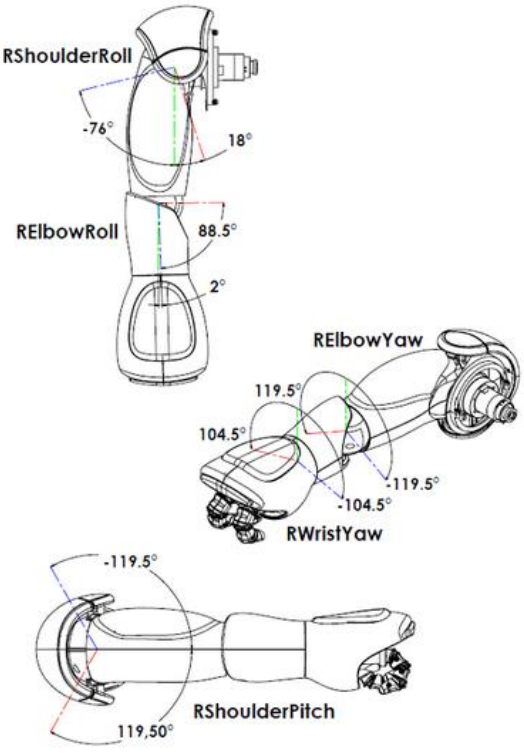


Figura 51. Límites y sentidos de giro de articulaciones en el brazo derecho de NAO [2].



3. Lenguaje de programación. NAO actualmente se puede programar en los siguientes lenguajes: Java, Java Script, C++, Python, MATLAB, Urbi y .Net. Aldebaran desarrolló un programa, Choregraphe, el cual permite a usuarios crear movimientos e interactuar con NAO de una manera fácil. Este consiste en unir bloques de código preprogramados para crear algún programa propio. Sin embargo, este software permite crear código propio en Python (ver Figura 53). La ventaja de utilizar Choregraphe es que este tiene un simulador (ver Figura 52 y Figura 54) que permite ver todos los movimientos y acciones de NAO antes de replicarlos en vida real. Este no solo replica los movimiento, sino que provee detalles como las posiciones actuales de cada articulación (ver Figura 55) [3].

Figura 52. Software de programación Choregraphe.

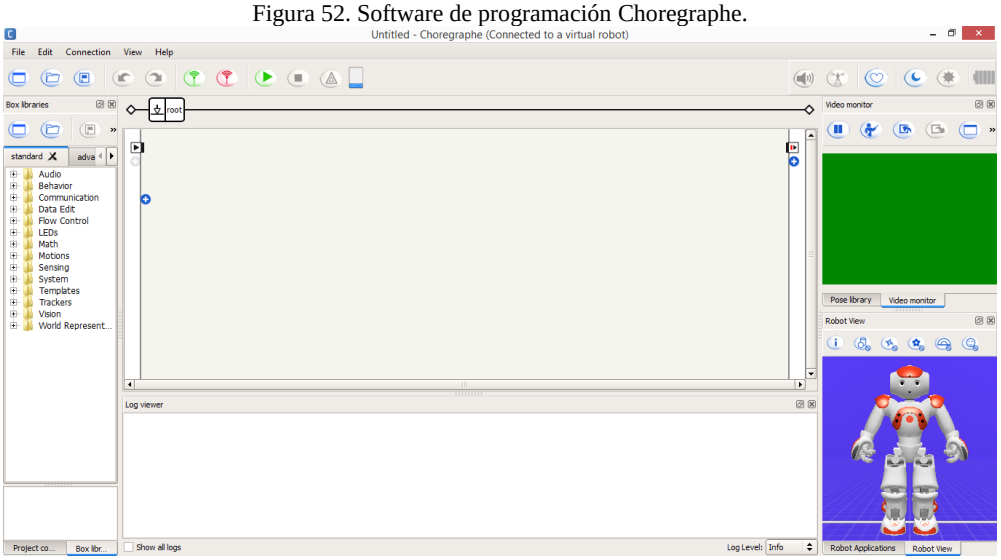


Figura 53. Bloque que incluye el formato necesario para programar a NAO en Python [2].

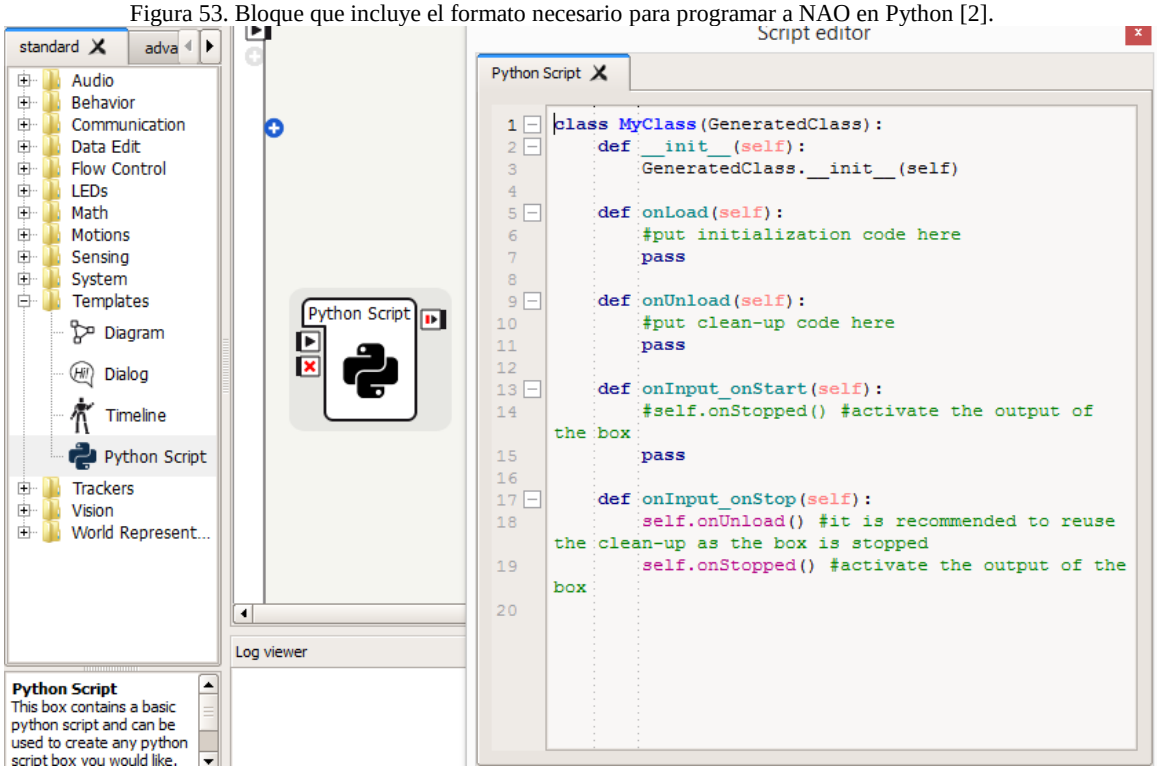


Figura 54. Simulación de NAO.

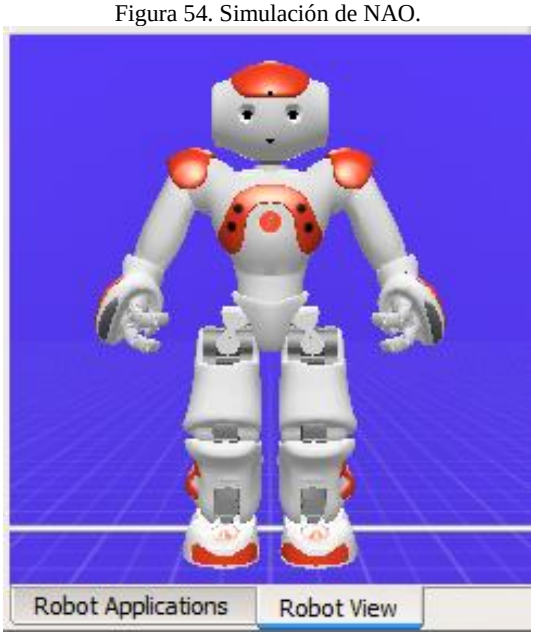
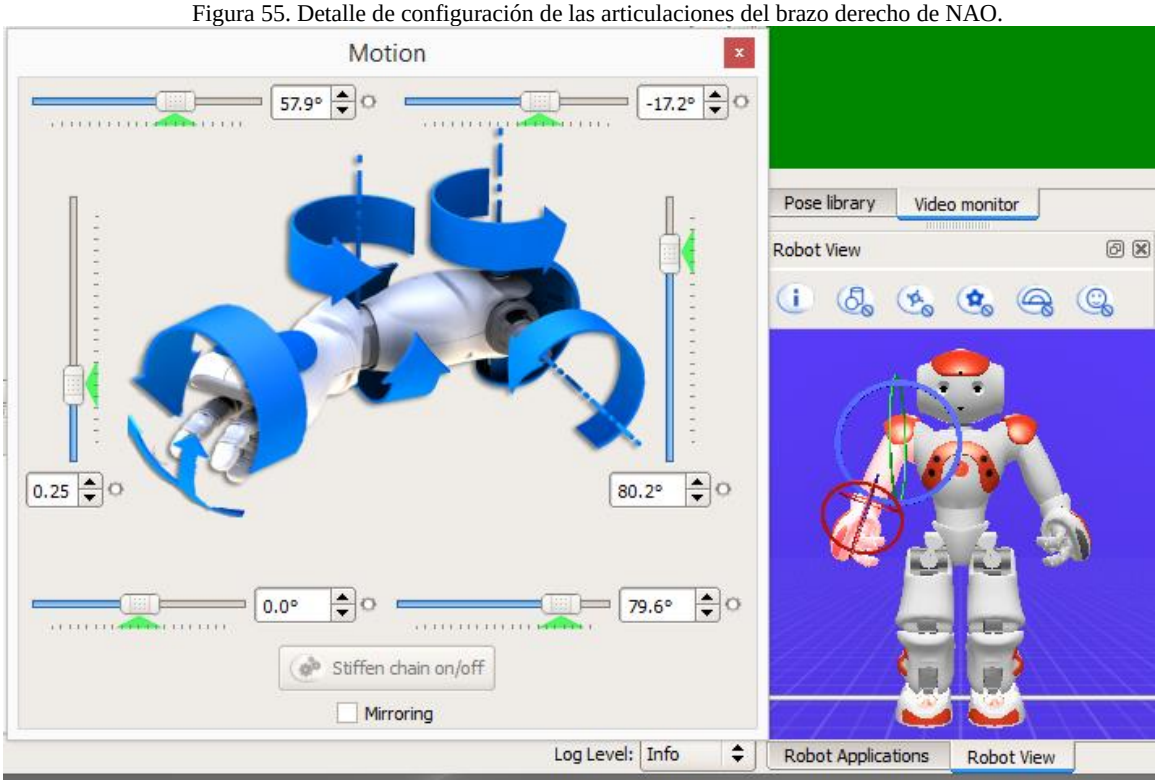


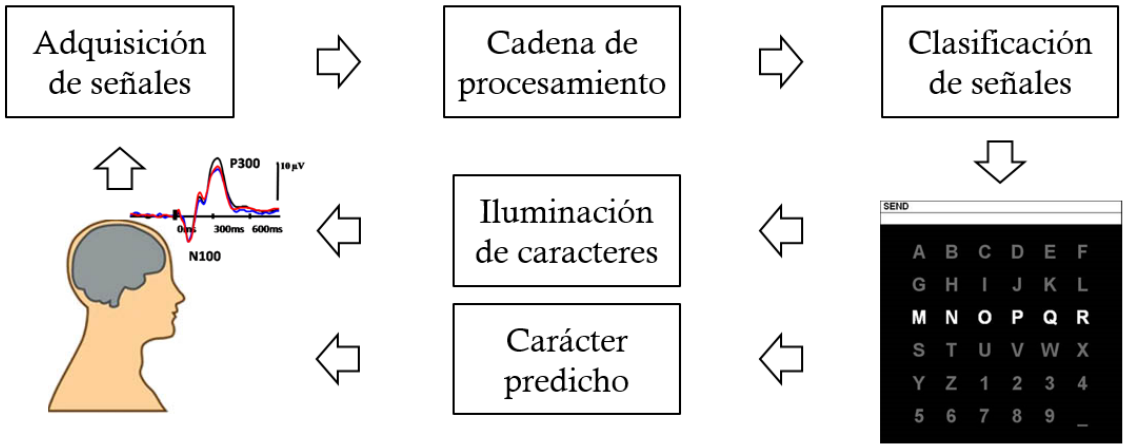
Figura 55. Detalle de configuración de las articulaciones del brazo derecho de NAO.



V. INTERFAZ CEREBRO COMPUTADORA

Como parte del presente megaproyecto, se realizó la implementación de un teclado P300, el cual, como ya se presentó anteriormente, opera utilizando el potencial relacionado a eventos conocido como onda P300. Para implementar dicho tipo de sistema, se deben de realizar las siguientes tareas: Primero, se debe de estimular al usuario a través de la presentación, en una pantalla de computadora, de una matriz con letras cuyos elementos son iluminados. Mientras esto ocurre, las señales cerebrales del paciente son adquiridas utilizando un dispositivo adecuado. Dichas señales son posteriormente procesadas a través de una cadena de procesamiento para mejorar su relación de señal a ruido y así permitir que un programa de computadora sea capaz de clasificarlas como señales que contienen, o no, a una onda P300. Con esta información disponible, se podrá determinar la letra en la cual el usuario estaba concentrado a lo largo de las iluminaciones, resultado que le será retroalimentado a través de la impresión del carácter predicho en pantalla. Las anteriores tareas son representadas como un diagrama de bloques en la Figura 56.

Figura 56. Diagrama de bloques de un teclado P300



Al momento de plantear la realización de este trabajo, se dividió al sistema del teclado P300, diagramado en la Figura 56, en dos módulos, los cuales serían trabajados por dos personas distintas. El primero de estos módulos se encargaría de implementar una aplicación de computadora capaz de generar la interfaz gráfica necesaria para estimular al usuario bajo un paradigma de filas y columnas del teclado P300, así como de decidir un equipo para adquirir de las señales cerebrales del usuario. Dicho módulo recibió el nombre de módulo de obtención de señales EEG y desarrollo de aplicación para interfaz cerebro computadora.

El segundo módulo, por su parte, se encargaría de tomar las señales adquiridas por el primero y de aplicarles una cadena de procesamiento implementar una cadena de procesamiento. Además, este módulo se encargó de la selección de un algoritmo de aprendizaje automático para ser utilizado como un clasificador de

las señales de interés, permitiendo así la predicción del carácter en el cual el usuario estaba concentrado. Este resultado (la letra predicha), era comunicado de vuelta al primer módulo para que lo mostrara en pantalla. Este segundo módulo fue llamado el módulo de procesamiento y clasificación de señales para interfaz cerebro computadora.

A continuación, se procederá a presentar el trabajo realizado para cada uno de los dos módulos anteriores, para finalizar mostrando su integración en un sistema de teclado P300 funcional.

A. MÓDULO DE PROCESAMIENTO Y CLASIFICACIÓN DE SEÑALES PARA INTERFAZ CEREBRO COMPUTADORA

1. Adquisición de señales de EEG.

Como se presentó anteriormente, la interfaz cerebro computadora de este trabajo fue dividida en dos módulos: Uno de dichos módulos tiene por objetivo estimular al paciente y capturar las señales de EEG producidas por este, mientras que el otro módulo, al cual corresponde esta sección, será utilizado para procesar las señales y clasificarlas adecuadamente. Por consiguiente, la adquisición de señales de EEG escapa del alcance de este módulo.

A pesar de lo anterior, para que un módulo sea capaz de procesar y clasificar señales de EEG, debe de tener disponibles dichas señales. Por tanto, con el ánimo de que ambos módulos pudieran ser desarrollados en paralelo, se buscaron plataformas alternativas que permitieran adquirir señales de EEG correspondientes a un teclado P300. Esto implica no solo el tener un dispositivo que permita medir las señales de EEG, sino también requiere el tener un sistema que sea capaz de estimular al paciente. A continuación, se presentan las alternativas halladas para obtener señales de EEG con las cuales poner a prueba el sistema de procesamiento y clasificación.

a. Señales de EEG obtenidas de bases de datos.

Algunos laboratorios que cuenta con el equipo adecuado para capturar señales de EEG colocan a disposición del público ciertas grabaciones realizadas. Por otra parte, también existen competencias en línea que tienen por objetivo validar diferentes técnicas de procesamiento de señales y métodos de clasificación en el contexto de las interfaces cerebro computadora [8]. Una de dichas competencias es la *BCI Competition* (de las palabras en inglés para “Competencia BCI”), en la cual se otorga a los participantes un conjunto de grabaciones realizadas sobre una interfaz cerebro computadora específica, para que ellos propongan un método que permita clasificar correctamente dichas señales.

Durante la segunda edición de esta competencia, se incluyó un conjunto de datos correspondientes a la aplicación del teclado P300 en su paradigma de filas y columnas [8]. Dicho conjunto de datos fue otorgado por el centro Wadsworth, el cual pertenece al departamento de salud del estado de Nueva York. Tal base de

datos estaba conformada por tres sesiones de grabaciones de EEG, todas realizadas sobre el mismo paciente Cada una de las sesiones estaban conformadas por cinco o seis palabras, siendo cada una de las palabras de entre tres y cinco letras. Las mediciones fueron realizadas utilizando un equipo que muestreaba las señales de EEG utilizando 64 electrodos y una frecuencia de muestreo de 240 Hz [8]. La estimulación del usuario fue realizada a una frecuencia de 5.7 Hz, siendo la duración de las iluminaciones de 100 ms y el intervalo inter-estimulo, de 75 ms. Además, los doce elementos de la matriz (filas y columnas) fueron iluminados un total de 15 veces cada una. Al finalizar las 15 iluminaciones de toda la matriz, se deba un intervalo de 2.5 s para indicar al usuario que se iniciaría la estimulación para la siguiente letra [8].

Además de incluir las señales EEG correspondientes a los 64 canales durante la duración de cada una de las grabaciones, el conjunto de datos también contenía varias listas que eran indispensables para poder dar una interpretación adecuada a las señales de EEG. Dentro de dichas listas, se encontraba una que indicaba cuando existía una fila o columna iluminada en la matriz, información dentro de la cual también se incluía un código de identificación de la fila o columna que se hallaba iluminada y sí dicho elemento correspondía a la letra en la cual el usuario estaba, teóricamente, concentrado. Se debe de recordar que, por ser estos datos unos de entrenamiento, se conoce de antemano la letra en la cual está concentrado el usuario para así poder etiquetar correctamente los datos de ejemplo que servirán para alimentar al algoritmo clasificador. Finalmente, también se incluía una lista de la cual se podía obtener el momento en el cual se iniciaba la estimulación correspondiente a una nueva letra [8].

Como ya se mencionó anteriormente, toda esta información es de suma importancia para poder analizar las señales de EEG correctamente. Como primer punto, debe de conocerse con exactitud el momento en que la iluminación de una letra inició ya que será dentro de una ventana de tiempo posterior a dicho estímulo que se podrá apreciar la onda P300. Además, es necesario conocer la fila o columna a que dicha iluminación correspondía, ya que de esta forma se podrá, posteriormente, agrupar las respuestas correspondientes a la misma fila o columna y promediarlas para mejorar la relación de señal a ruido de la onda P300. De igual manera, es necesario conocer cuáles estimulaciones corresponden a una letra y cuáles a otra, con el objetivo de no mezclar las señales de EEG correspondientes a dos situaciones distintas. Finalmente, en esta etapa de entrenamiento, es fundamental conocer cuáles respuestas corresponden al estímulo objetivo y cuáles a estímulos no objetivos, ya que, de lo contrario, se ingresará información incorrecta al clasificador y este no podrá realizar una identificación adecuada de las nuevas señales que reciba, puesto que los ejemplos con los cuales se alimentó estaban erróneamente etiquetados.

A continuación se presentarán las líneas de código en MATLAB con las cuales se obtuvieron las señales de ejemplo para poder procesarlas y, posteriormente, alimentar al clasificador. Antes de continuar, se debe de tener claro que el objetivo de esta tarea es la de separar las señales de EEG en segmentos correspondientes a un intervalo de tiempo posterior a la presentación de un estímulo, segmentos que estarán, cada uno,

identificados como correspondiente a un estímulo objetivo o no. Además, en la explicación que sigue, se tendrán a las siguientes listas disponibles, puesto que, como ya se explicó anteriormente, eran parte del conjunto de datos obtenido. Todas estas listas corresponden a vectores que cuentan con un elemento por cada muestra de las señales de EEG capturadas; i.e., estos vectores contiene 240 elementos por cada segundo de grabación:

- *Flashing* tendrá un valor de 1 cuando exista una fila o columna iluminada. En cualquier otro caso, su valor será de 0.
- *PhaseInSequence* tendrá un valor de 1 mientras no se haya iniciado las estimulaciones correspondientes a una letra; 2, cuando se están aplicando los estímulos a través de la iluminación de filas y columnas; y 3, cuando finalizan las estimulaciones correspondientes a la letra. Esto quiere decir que, cada vez que el valor de esta lista cambia a 3, se está realizando la estimulación para una nueva letra [8].
- *StimulusCode* contiene el código de la fila o columna que se encuentra actualmente iluminada. Dicho código corresponde a un número entero que va del 1 al 12. Si ninguna iluminación está ocurriendo, esta lista tomará el valor de cero.
- *StimulusType* indica si la iluminación actual corresponde a la letra en la cual el usuario debería de estar concentrado. En dicho caso, toma un valor de 1; en cualquier otro caso, de 0.

Ya con las anteriores listas definidas, el código que permite recuperar las señales de EEG en segmentos de tiempo posteriores a la estimulación y su respectiva clasificación, es el siguiente:

```
%Se hallan los números de muestra en los cuales se cambió de letra.
%Lista en la cual se almacenarán los números de muestra en los cuales
%ocurrió un cambio de letra:
cambioDeLetra = [ ];
%Se hallan los posibles momentos en los cuales ocurrió el cambio de
%letra:
posibleCambioDeLetra = find(PhaseInSequence==3);
indiceAnterior = -100;
for i = 1 : size(posibleCambioDeLetra,1)
    %Se encuentran aquellos momentos que no son precedidos por otra
    %muestra con un PhaseInSequence igual a 3, lo cual indica que se
    %ha cambiado de letra, por lo que se almacenan dichos números de
    %muestra:
    indiceActual = posibleCambioDeLetra(i);
    if (indiceActual - indiceAnterior ~= 1)
        cambioDeLetra = [cambioDeLetra; indiceActual];
    end
    indiceAnterior = indiceActual;
end
```

```
%Se hallan los números de muestra en los cuales inició una estimulación.
%En esta matriz se almacenarán las señales de EEG, como vectores fila,
%correspondientes a diferentes estimulaciones:
X = [ ];
%Mientras que este vector contendrá la clasificación de cada una de
%las estimulaciones: 1 si corresponden a un estímulo objetivo o 0 si
%no.
y = [ ];
codigo = [ ]: %Y esta lista contiene los códigos de las estimulaciones.
%Se hallan todos los momentos en los cuales había una iluminación y
%aquellos momentos en los cuales había una iluminación cuyo número de
%muestra no correspondía al número de muestra siguiente al de la
%iluminación anterior era un nuevo estímulo:
posibleInicioEstimulo = find(Flashing == 1);
indiceAnterior = -100;
for i = size(posibleInicioEstimulo, 1)
    indiceActual = posibleInicioEstimulo(i);
    if (indiceActual – indiceAnterior ~= 1)
        datos = signal(indiceActual : indiceActual + constante, :);
        %En dicho caso, se almacenaban las señales de EEG desde la
        %muestra actual hasta un número fijo de muestras después en
        %la matriz X, así como la clasificación de dicha estimulación
        %en el vector y, y su código de identificación:
        datos = datos(:);
        X = [X; datos];
        y = [y; StimulusType(indiceActual)];
        codigo = [codigo; StimulusCode(indiceActual)]
    end
    indiceAnterior = indiceActual;
end
```

Al finalizar la ejecución del código anterior, se obtendría una matriz *X* y los vectores *y*, *código* y *cambioDeLetra*, los cuales contienen la información suficiente para poder dar una interpretación correcta a los segmentos de señales EEG adquiridos, procesarlos y, luego de eso, dividirlos en un conjunto de entrenamiento, validación cruzada y prueba con los cuales entrenar y reportar el rendimiento del algoritmo clasificador.

b. Señales de EEG obtenidas utilizando el dispositivo Emotiv EPOC. A pesar de que las bases de datos con señales de EEG grabadas en una sesión de teclado P300 ofrecían una gran ayuda para poder obtener elementos con los cuales probar al sistema, presentaban la desventaja de que no se contaban con más grabaciones del mismo sujeto para seguir probando el sistema. Además, una desventaja todavía mayor era que, puesto que no se contaba con el mismo dispositivo para capturar las señales de EEG ni con el sujeto sobre el cual se realizaron las grabaciones utilizadas en el algoritmo clasificador, era imposible probar el sistema en tiempo real, sino solamente fuera de línea.

Ahora bien, el problema con los dispositivos de adquisición de EEG es que estos son, en general, grandes, no portátiles y de un costo elevado, además de que utilizan conductores para transmitir las señales capturadas

[20]. Estas características hacen de un equipo de EEG un recurso costoso y de las grabaciones con dicho equipo un procedimiento poco flexible que debe de ser realizado en un mismo lugar, restringiendo la posibilidad de utilizar una interfaz cerebro computadora en ambientes cotidianos para la persona que la utiliza.

Una alternativa frente a los equipos tradicionales de adquisición de EEG es el Emotiv EPOC, un dispositivo comercialmente disponible distribuido por la empresa Emotiv [21] y presentado previamente en el marco teórico. Dicha herramienta se caracteriza por ser una portátil, ya que cuenta con un diseño compacto, una batería como fuente de alimentación interna y, además, transmite las señales capturadas de forma inalámbrica. Estas propiedades facilitan el uso del dispositivo en diferentes ubicaciones, además de que facilitan la instalación de este mismo, puesto que no debe de lidiarse con un gran número de conexiones. Otra característica positiva del Emotiv EPOC es su costo relativamente reducido. La importancia de dicha propiedad radica en que, si se desea replicar la interfaz cerebro computadora para su uso en diferentes pacientes, la viabilidad de dicho proyecto estará dada por el costo que implica cada una de las interfaces. Si cada una de las interfaces requiriese la compra de un dispositivo de adquisición de EEG tradicional, el costo monetario sería un factor prohibitivo para llevar a cabo el proyecto. Pero, al utilizar herramientas de bajo costo, la accesibilidad económica de cada una de cada una de estas interfaces cerebro computadora se verá incrementada. Su bajo costo económico y facilidad de transporte, junto al hecho de que este dispositivo se encontraba disponible dentro de la universidad, fueron las razones fundamentales por las cuales se eligió esta alternativa frente a otras.

Como ya se ha presentado en el marco teórico, el dispositivo Emotiv transmite las muestras capturadas a una computadora utilizando una comunicación *bluetooth.* Además, los datos transmitidos por el Emotiv EPOC se encuentran encriptados, por lo que, para su correcta interpretación, es necesario utilizar ciertos programas específicos. Asimismo, como ya se presentó previamente, para verificar el adecuado funcionamiento del Emotiv EPOC, puede utilizarse el programa "Emotiv EPOC" incluido con el dispositivo. En dicho programa, el color en cada una de las ubicaciones de los electrodos indica la calidad de las señales obtenidas por ellos y puede variar entre negro, rojo, naranja, amarillo y verde, indicando el color negro una calidad nula y cada color una calidad cada vez mayor, hasta llegar al verde que representa el óptimo. Se recomienda que, cada vez que se utilice el dispositivo Emotiv EPOC, se abra este programa para verificar que todas las ubicaciones presentan al menos un color amarillo o, en el mejor de los casos, verde.

1) Obtención de señales del Emotiv EPOC a través de los programas BCI2000 y FieldTrip.

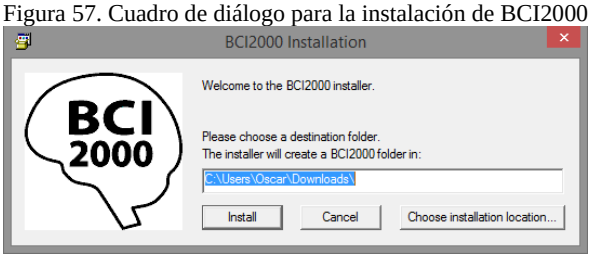
Una de las aplicaciones que presenta la capacidad de recuperar correctamente las señales transmitidas por el Emotiv EPOC es BCI2000. BCI2000 es un programa de uso general para interfaces cerebro computadora, desde el cual se pueden generar estímulos y recibir datos provenientes de diferentes dispositivos de adquisición de EEG. Dicho programa es una creación del laboratorio Schalk (el cual es parte del centro Wadsworth del departamento de salud del estado de Nueva York) y se encuentra disponible de manera

gratuita para propósitos educativos y de investigación sin ánimos de lucro [81]. Para instalar dicho programa y poder recuperar las señales de EEG provenientes del dispositivo Emotiv EPOC, deben de seguirse los pasos descritos a continuación.

a) Instalación y configuración de BCI2000. Para obtener los programas ejecutables que servirán para la instalación de BCI2000, es necesario primero crear una cuenta de usuario dentro del sitio web de dicho programa. Para ello, el primer paso es dirigirse al sitio http://www.bci2000.org/GPL/, en el cual se presenta el acuerdo de licencia para el uso del programa. Una vez leído dicho acuerdo, se procede a dar clic en el botón de "*I Agree*" (del inglés para "Estoy de acuerdo"), con lo cual se nos llevará a una nueva página. En esta página, se solicita el nombre de cuenta deseado, así como datos de identificación del usuario. Una vez ingresados todos los campos requeridos, se presiona el botón de "*Create account*" (del inglés para "Crear cuenta"). Luego de realizar esto, se recibirá un correo electrónico en la dirección especificada entre los datos ingresados anteriormente, en la cual se presentará el nombre de usuario y la contraseña que se tendrá para poder acceder a los archivos de BCI 2000.

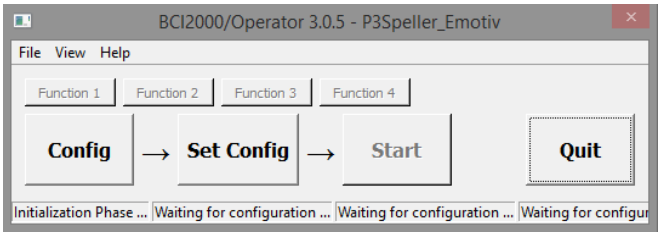
Al contar con el nombre de usuario y contraseña, se debe de dirigir al sitio http://bci2000.org/downloads/bin/BCI2000Contrib.exe. Al ingresar a dicho sitio, el explorador mostrará un mensaje solicitando un nombre de usuario y una clave, campos en los cuales se debe de ingresar la información recibida por correo previamente descrita. Una vez dichos datos son validados, se dará la opción de descargar un archivo ejecutable de nombre BCI2000Contrib.exe. Se almacena dicho archivo en una ubicación deseada y posteriormente, se ejecuta. Al ejecutar dicho archivo, se abrirá un cuadro de diálogo donde se solicita al usuario que indique la ubicación en la cual desea instalar BCI2000. Dicho cuadro de diálogo se presenta en la Figura 57. El usuario puede elegir la ubicación que considere más adecuada para la instalación y, luego de ello, debe de presionar el botón de *"Install"* (del inglés para "Instalar"). Luego de ello, se debe de esperar un tiempo a que finalice la instalación de BCI 2000. Una vez esto ocurra, se mostrará un mensaje que indica que el programa ha sido correctamente instalado en la ubicación indicada. Al dirigirse a dicha ubicación, se encontrará un conjunto de carpetas que conforman al programa y cuya compresión será de suma utilidad. Dentro de las carpetas más importantes, se tiene la carpeta "*batch"*, desde la cual se puede ejecutar archivos .bat que inicializan al programa de BCI2000 con cierto módulo de presentación de estímulos, adquisición de las señales y procesamiento de estas ya cargados. La carpeta "*data*" por su parte, contendrá la información grabada por BCI2000, la cual se almacena en archivos que tienen una extensión .dat, mientras que la carpeta "*parms*" almacenará las configuraciones correspondientes a diferentes experimentos. La carpeta "*program*" contiene los programas que permiten la obtención de señales de EEG desde diferentes equipos de adquisición y "*tools*" incluye múltiples herramientas que permiten analizar los datos adquiridos con BCI2000 o re-expresarlos en otros formatos, para que puedan ser interpretados por otros programas. Finalmente, se cuenta con una carpeta que contiene documentación del programa, llamada "*doc*".

Figura 57. Cuadro de diálogo para la instalación de BCI2000



Para poder adquirir datos desde el Emotiv EPOC utilizando BCI2000 y un teclado P300, se debe de dirigir a la carpeta "*batch*" y realizar una copia del archivo P3Speller_SignalGenerator.bat, colocándole a dicha copia el nombre de P3Speller_Emotiv.bat. Luego, se abre dicho archivo para su edición y la línea que reza "*Start executable SignalGenerator --local*" debe de ser modificada a "*Start executable Emotiv --local*". Luego, se guarda el archivo y se cierra el editor, para posteriormente, dar doble clic sobre el archivo .bat. Al hacer esto, se ejecutará el programa BCI2000, ya preparado para recibir señales provenientes del Emotiv EPOC. La pantalla correspondiente al programa BCI2000 se muestra en la Figura 58.

Figura 58. Cuadro de diálogo principal de BCI2000



Antes de que BCI2000 pueda dar una interpretación correcta a las señales provenientes del Emotiv EPOC, deben de aplicarse unas configuraciones más. Para ello, debe de abrirse la ventana de configuración dando clic sobre el botón "*Config*". En el cuadro de diálogo abierto, se debe de seleccionar la pestaña de *"Source"* (del inglés para "Origen") y colocar los siguientes datos en los campos correspondientes [52]:

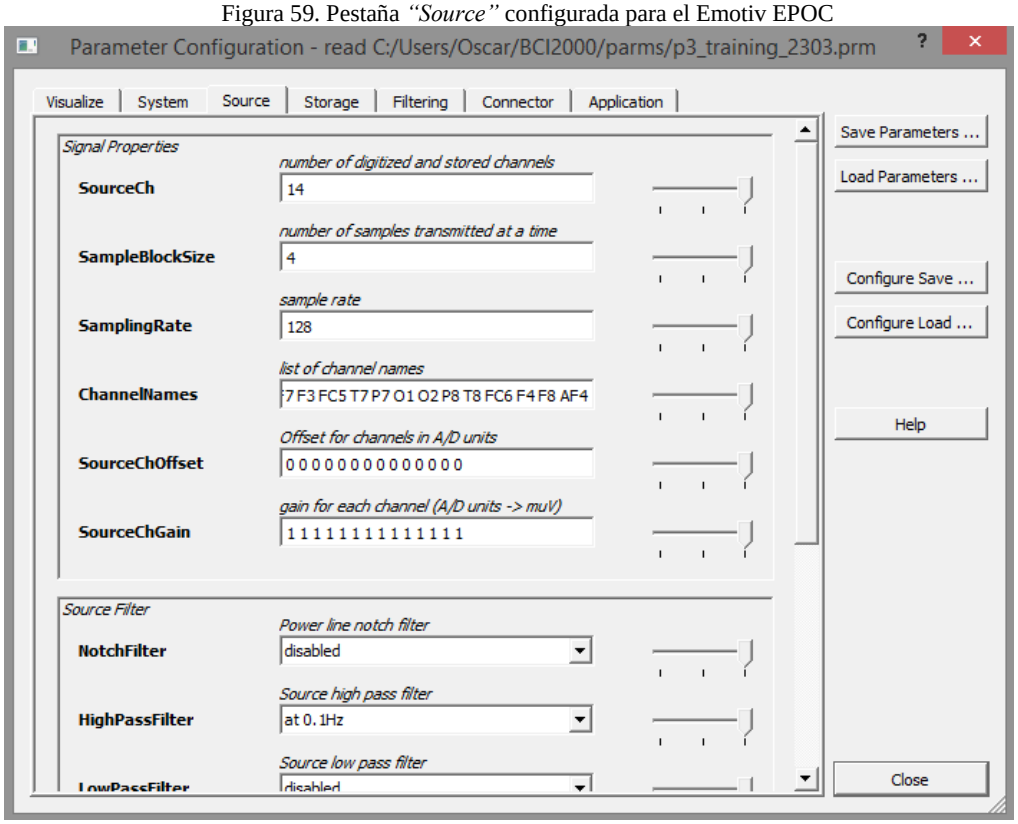
- *SourceCh* debe de tener un valor de 14, puesto que son 14 canales los que ofrecen mediciones de señales EEG en el Emotiv EPOC.
- *SampleBlockSize* debe de tener un valor de 4.
- *SamplingRate* debe tener un valor de 128, correspondiente a la frecuencia de muestreo de cada uno de los canales, en Hz.
- *ChannelNames* corresponde a los nombres de los canales del Emotiv EPOC, los cuales vienen dados por su nombre en el sistema internacional 10-20. Por tanto, en esta casilla debe de escribirse "AF3 F7 F3 FC5 T7 P7 O1 O2 P8 T8 FC6 F4 F8 AF4".
- *SourceChOffset* indica el valor medio de cada una de las señales medidas en los canales. Si bien el Emotiv EPOC añade una constante a cada uno de los canales para facilitar su trazado gráfico, dichos valores serán retirados utilizando un filtro pasa-altas, el cual se configurará en una de las opciones

posteriores [52]. Por consiguiente, en este campo se debe de escribir "0" por cada canal en el dispositivo.

- *SourchChGain* es una lista que contiene los factores para convertir las unidades del convertidor analógico-digital (ADC) ofrecidas por el dispositivo en µV. Los valores dados por el Emotiv EPOC ya se encuentran en µV [52], por lo que en este campo se debe de colocar "1" por cada canal en el dispositivo.
- *HighPassFilter* indica un filtro pasa-altas que será aplicado por BCI2000 y se encuentra en la sección *"SourceFilter"* (del inglés para "Filtro de la fuente"). Para poder utilizar el Emotiv EPOC con BCI2000, es obligatorio elegir el filtro pasa-baja a 0.1 Hz.
- *TransmitChList* es un campo que se encuentra en la sección de "*Online Processing"* (del inglés para "Procesamiento en línea") e indica los canales que serán considerados durante el procesamiento de señales. Si se desea que los 14 canales sean tomados en cuenta durante el procesamiento, en este campo deben de escribirse los número del 1 al 14 separados por un espacio.

Una vez aplicadas estas modificaciones, la pestaña de "*Source*" debería de verse como se presenta en la Figura 59.

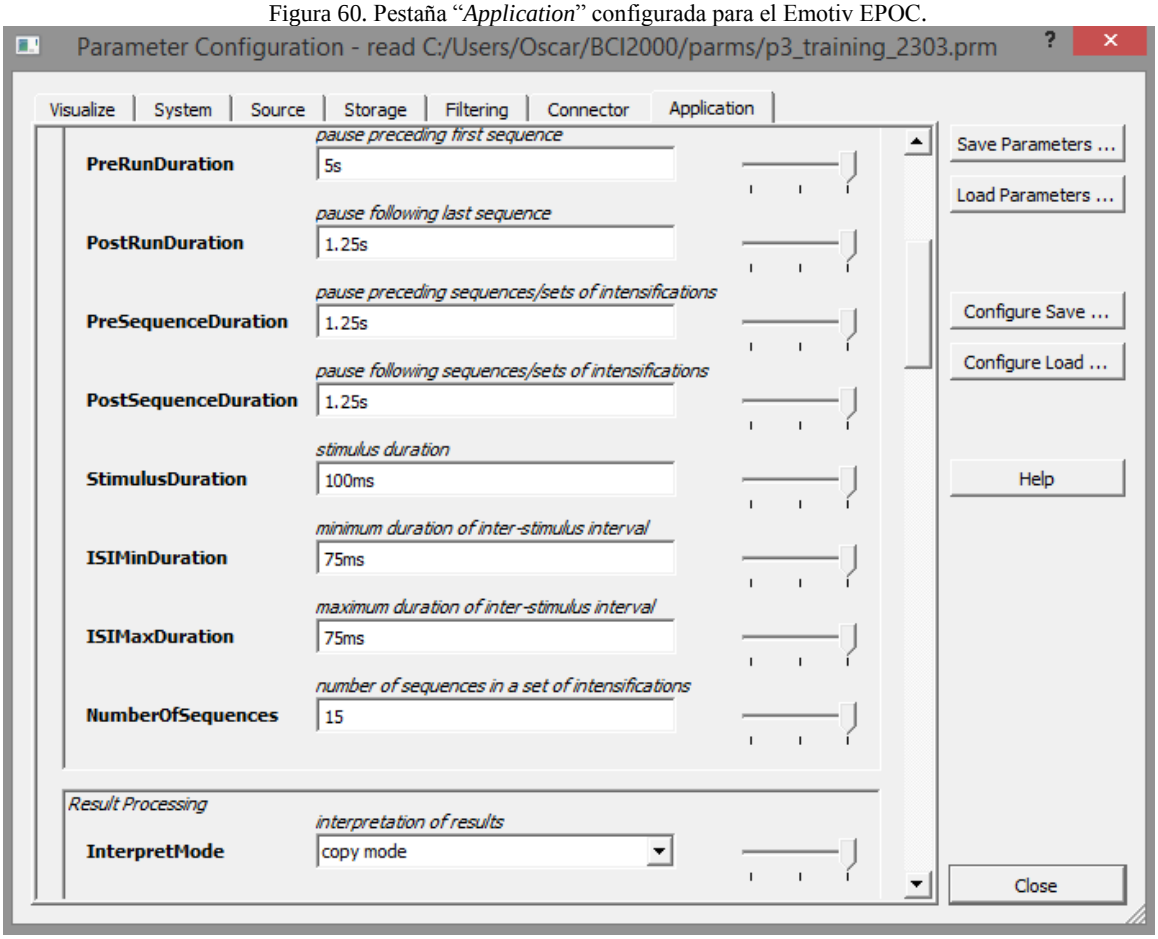
Figura 59. Pestaña *"Source"* configurada para el Emotiv EPOC



En cuanto a las características de la estimulación del teclado P300, estas pueden ser consultadas y modificadas en la pestaña de "*Application*" (del inglés para "Aplicación"), la cual se ilustra en la Figura 60.

Dentro de la sección de "*Sequencing*" (del inglés para "Secuenciación"), se pueden modificar diversos parámetros relacionados con los tiempos de estimulación, tales como la duración de cada una de las estimulaciones, del intervalo inter-estímulo, el tiempo entre el final de las iluminaciones correspondientes a una letra y el inicio de otra, y el número de veces que se repetirá la iluminación de toda la matriz. También se puede indicar el tipo de grabación que se realizará en la sección de "*Result Processing"* (del inglés para "Procesamiento del resultado"). Esta puede ser "*Copy Mode*", el cual corresponde a las secciones de entrenamiento puesto que el programa conocerá las letras que serán ingresadas por el usuario, u "*Online Free Mode*", el cual corresponde a un procesamiento en línea en el cual el usuario elige independientemente las letras que desea escribir, sin que el sistema tenga conocimiento de ellas. En el caso se elija el "*Copy Mode*", la palabra a ser escrita por el usuario puede ingresarse en el campo "*TextToSpell*" (del inglés para "Texto a escribir") que se encuentra en el campo de *"Speller"* (del inglés para "Deletreador").

Figura 60. Pestaña "*Application*" configurada para el Emotiv EPOC.



Con el objetivo de no repetir el ingreso de estas configuraciones cada vez que se inicia el programa BCI2000, estas mismas pueden ser grabadas. Dentro de la ventana abierta para la configuración, se puede hacer clic sobre el botón "*Configure Save…*" (del inglés para "Configurar grabado…"), lo cual abrirá un cuadro de diálogo dentro del cual se pueden seleccionar los parámetros de la configuración a grabar. Una vez

configurado esto, se procede a hacer clic sobre el botón "*Save Parameters…*" (del inglés para "Grabar parámetros…"), con lo cual se grabarán las configuraciones indicadas anteriormente en un archivo de extensión .prm. Se recomienda que dichos archivos se guarden dentro de la carpeta "*parms*" de BCI2000. Para cargar las configuraciones grabadas, primero se debe de hacer clic sobre el botón "*Configure Load…*" (del inglés para "Configurar carga…") y seleccionar los parámetros que se desean cargar. Luego, se hace clic sobre el botón de "*Load Parameters…*" (del inglés para "Cargar parámetros…") y se selecciona el archivo .prm del cual se desean recuperar los parámetros.

Una vez se finaliza con la configuración de BCI2000, se debe de hacer clic sobre el botón *"Close"* (en español, "Cerrar"), y luego, en la ventana principal de BCI2000, se debe de hacer clic sobre "*Set Config*" (del inglés para "Aplicar configuración"). Antes de presionar dicho botón, se debe de verificar que el *dongle* del Emotiv EPOC ya se encuentra conectado a la computadora a utilizar y que dicho dispositivo está encendido. Luego de presionado dicho botón, se abrirán varias ventanas, siendo las de mayor interés aquellas que contienen las señales recibidas desde el Emotiv EPOC y la matriz del teclado P300 para estimular al paciente. Después, se puede iniciar una sesión de teclado P300 presionando el botón de *"Start"* (en español, "Empezar") presentado en la pantalla principal de BCI2000.

b) Herramientas de BCI2000. BCI2000 presenta un conjunto de herramientas que hacen de este programa uno muy versátil, al expandir las acciones que pueden realizarse con él. Para este trabajo estaremos interesados en dos de las herramientas incluidas con BCI2000. La primera de ellas es el programa *load_bcidat*, el cual se encuentra en la dirección "*BCI2000>tools>mex*". Dicho programa permite extraer la información contenida en los archivos .dat dentro de un programa de MATLAB, para luego poder trabajar con tales datos en el entorno de MATLAB. En el caso de estimulaciones utilizando el teclado P300, los contenidos del archivo .dat son la señal de EEG y las mismas listas que se presentaron anteriormente en la sección de señales de EEG obtenidas de bases de datos.

El programa de *load_bcidat* se encuentra disponible, dentro de la ruta especificada, como un código de MATLAB .m, así como en versiones compiladas para diferentes arquitecturas. Se recomienda utilizar las versiones compiladas, puesto que disminuirán el tiempo de ejecución del programa. Además, otro cuidado que se debe de tener al momento de utilizar este programa es que los datos obtenidos deben de ser convertidos a números enteros de precisión doble para poder operarlos adecuadamente. Dicha conversión se realiza utilizando la siguiente línea de código en MATLAB:

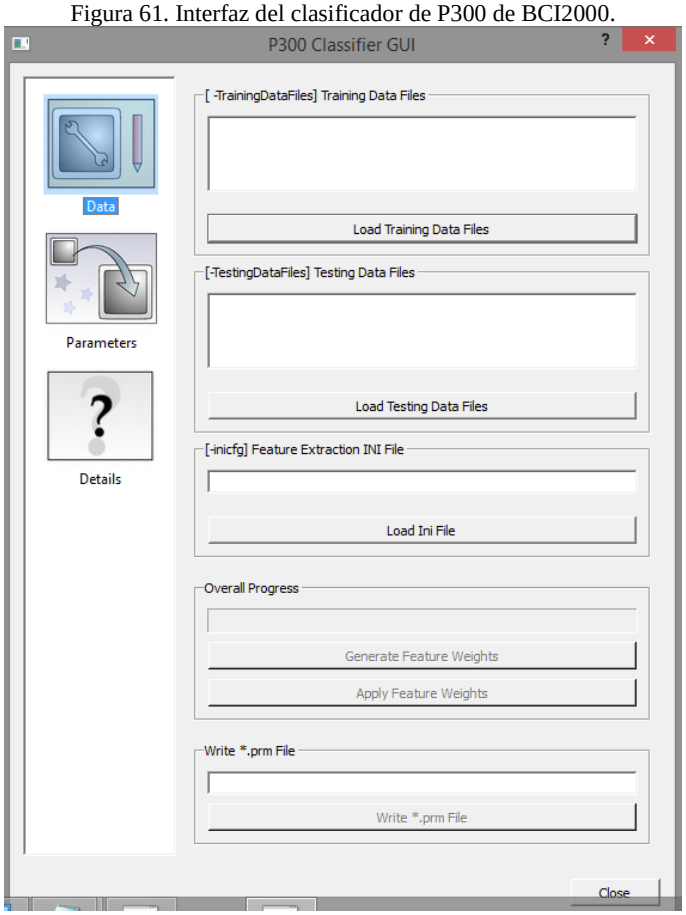
```
arreglo = double(arreglo);
```

Donde arreglo corresponde a una matriz o lista que contiene información numérica.

La segunda herramienta de BCI2000 de interés es un programa que permite el entrenamiento de un algoritmo clasificador para un teclado P300. Dicho programa tiene el nombre de *P300Classifier.exe* y se

encuentra en la dirección "*BCI2000>tools>P300Classifier*". Al abrir esta herramienta haciendo doble clic sobre ella, se obtiene la ventana mostrada en la Figura 61.

Figura 61. Interfaz del clasificador de P300 de BCI2000.



En dicha ventana, pueden seleccionarse los archivos .dat a utilizar como datos de entrenamiento y de validación del clasificador, así como indicar los parámetros del clasificador en la pestaña *"Parameters"*. El entrenamiento puede ejecutarse dando clic en el botón "*Generate Feature Weights*" (en español, "Generar pesos de las características") y los resultados pueden almacenarse usando el botón "*Write *.prm File*" (del inglés para "Escribir el archivo *.prm"). Una vez almacenados dichos parámetros, estos pueden cargarse en la configuración del programa BCI2000 descrita anteriormente (la cual se abre al dar clic sobre el botón *"Config"* de la ventana principal de BCI2000), de tal manera que, al realizar una nueva sesión de teclado P300, el clasificador se hallará ya ajustado para el usuario con cuyos datos se realizó el entrenamiento. Cabe mencionar que el clasificador utilizado por esta herramienta es un SWLDA.

Las dos herramientas de BCI2000 presentadas son de sumo interés para este trabajo por las siguientes razones. La primera herramienta permitirá la manipulación de los datos, generados por BCI2000 y capturados por el Emotiv EPOC, en el entorno de MATLAB, en el cual se llevarán a cabo el procesamiento y

clasificación de las señales. De esta manera, se podrán adquirir y manipular datos obtenidos sobre sujetos propios a este estudio. Por otra parte, la segunda herramienta otorgará un procesamiento y clasificación ajenos a los realizados en este trabajo, por lo que el rendimiento de dicho clasificador servirá como una referencia para los resultados generados por este módulo. La comparación será válida puesto que ambos clasificadores serán alimentados con datos provenientes de los mismos sujetos y obtenidos por el mismo dispositivo de adquisición de EEG.

Sin embargo, existe una necesidad para el desarrollo de este módulo que todavía no es suplida por ninguna de estas dos herramientas: El poder realizar clasificaciones en tiempo real. Como ya se presentó, el programa de *load_bcidat* permite la extracción de información de los archivos .dat para ser utilizada en MATLAB. Desafortunadamente, dichos archivo son generados hasta finalizar una sesión del teclado P300, por lo que no puede usarse esta vía para acceder a los datos recopilados por BCI2000 mientras el programa corre. Para poder lograr esta otra acción, se debe de utilizar otro programa, conocido como *FieldTrip* [30], cuya instalación y uso se presentan a continuación.

c) Instalación y uso del programa *FieldTrip*. *FieldTrip* es un conjunto de herramientas para realizar análisis de señales de EEG y MEG (i.e., electromiografía, que corresponde a las señales eléctricas producidas por los músculos) que ha sido desarrollado por investigadores del Instituto Donders para el cerebro, cognición y comportamiento y el Instituto Max Planck, ambos ubicados en los Países Bajos [63]. En este caso, este programa nos interesará únicamente como un medio para comunicar en tiempo real a BCI2000 con MATLAB.

Para instalar a este programa, se debe de dirigir a la dirección web ftp://ftp.fcdonders.nl/pub/fieldtrip/. Desde aquí, se puede descargar el archivo .zip correspondiente a la versión más reciente del programa. Luego, basta con extraer dicha carpeta en la ubicación deseada para finalizar la instalación. Por otra parte, en la carpeta "*batch*" de BCI2000 debe de crearse una copia del archivo P3Speller_Emotiv.bat, pero esta vez dándole el nombre de P3Speller_Fieldtrip.bat. Luego, se debe de editar dicho archivo, cambiando la línea de "*Start executable P3SignalProcessing --local*" a "*Start executable FieldTripBuffer --local*" y se guardan los cambios realizados. De esta forma, al ejecutar P3Speller_Fieldtrip.bat, cargar y aplicar una configuración y, finalmente, presionar el botón de "Start", BCI2000 estará enviando datos a MATLAB a través de *FieldTrip*.

Para tener acceso a dichos datos desde MATLAB, primero se debe de añadir la ubicación de *FieldTrip* a la ruta de búsqueda de MATLAB. Para lograr esto, se puede utilizar el comando *addpath* de MATLAB. Luego, dentro del programa de MATLAB desde el cual se extraerá la información, debe definirse una variable con la dirección del puerto donde se encuentra la aplicación de *FieldTrip;* e.g, puede escribirse:

```
filename = 'buffer://localhost:1972';
```

Tal y como lo deja entrever la línea anterior, *FieldTrip* almacena los datos generados por BCI2000 en un búfer, para que estos puedan ser accedidos por la aplicación de MATLAB. Para obtener información sobre el estado actual del búfer, puede utilizarse el siguiente comando:

```
hdr = ft_read_header(filename);
```

De esta forma, en la variable hdr se tendrán datos tales como el número de muestra actual, la frecuencia de muestreo y el número de canales de los cuales se está obteniendo información. Cada vez que se requiera obtener estos datos actualizados, se debe de ejecutar de nuevo la función *ft_read_header* [9].

Por otra parte, si lo que se desea es recuperar las muestras capturadas por BCI2000 a través del dispositivo de adquisición de EEG, se debe de utilizar el siguiente comando:

```
dat = ft_read_data(filename, 'header', hdr, 'begsample', begsample,
                   'endsample', endsample, 'chanindx', chanindx);
```

Debe de notarse que, para poder utilizar la función *ft_read_data*, debió de utilizarse primero la función *ft_read_header* para poder contar con un encabezado hdr disponible. A la función *ft_read_data* también se le pueden indicar los parámetros opcionales *begsample*, *endsample* y *chanindx*, los cuales corresponden, respectivamente, al número inicial y final del intervalo de muestras que se quieren recuperar, así como al número de los canales de los cuales se quieren obtener las muestras [9].

Finalmente, para poder recuperar la información respectiva a las listas de estimulación presentadas anteriormente, se debe de utilizar el comando:

```
[arreglo] = ft_read_event(filename, 'type', 'NombreDelEvento');
```

Donde NombreDelEvento corresponde a una cadena de texto con la lista de interés, tales como *StimulusCode*, *PhaseInSequence* y *StimulusType*. El arreglo recuperado contiene dos listas, *arreglo.sample* y *arreglo.value*, las cuales contienen, respectivamente, el número de muestra en que ocurrió el evento (i.e., la iluminación) y su valor respectivo.

2) Obtención de señales del Emotiv EPOC a través del módulo de obtención de señales. En esta tercera alternativa, se incluyó el código de procesamiento y clasificación en línea de las señales P300 dentro del módulo de obtención de señales y desarrollo de aplicación para interfaz cerebro computadora de este mismo megaproyecto [14]. Dicho módulo se encargaba de generar la interfaz gráfica del teclado P300 y de manipularla para estimular al usuario, registrando las respuestas de este mismo utilizando un dispositivo Emotiv EPOC. Además, este módulo se encargaba de generar listas similares a las presentadas anteriormente para que el módulo de procesamiento y clasificación de las señales funcionara adecuadamente. Para más información sobre este módulo, favor consultar el trabajo escrito correspondiente.

Cuando se hable de la grabación de datos de EEG, sin importar cuál módulo de adquisición fue utilizado, se estarán utilizando los términos sesión y corrida para identificar la estructura con la cual los datos fueron obtenidos. Una sesión consiste en un intervalo de tiempo que inicia en el momento en el cual se le colocó al usuario el dispositivo de adquisición de señales de EEG y termina al momento de retirar dicho dispositivo de la cabeza del usuario. Por otra parte, una corrida corresponde a un intervalo de tiempo durante el cual el usuario fue estimulado y sus señales fueron grabadas, de forma continua. Esto pudo haber incluido la grabación de una o varias letras. Dadas las definiciones anteriores, se puede observar que una sesión puede englobar una o varias corridas, así como una corrida puede contener una o varias letras.

2. Procesamiento de señales de EEG. Una vez adquiridas las señales de EEG, es necesario aplicar uno o varios métodos que permitan mejorar la relación de señal a ruido de dichas señales, con el propósito de facilitar al clasificador la tarea de determinar tanto las características comunes a los elementos de una misma clase, como aquellas características que permiten diferencia a una clase de otra. Tal y como se definió en el marco teórico, a la secuencia de métodos aplicados sobre las señales previo a su clasificación, las definiremos como cadenas de procesamiento. Dichas cadenas de procesamiento se caracterizan no solo por los métodos que las componen, sino también por el orden en que estos son aplicados. De tal forma que, si dos cadenas están compuertas por las mismas etapas, pero estas son aplicadas en un orden distinto, no son consideradas como cadenas iguales.

En el presente capítulo, se presentarán distintos elementos que pueden utilizarse dentro de una cadena de procesamiento, pero sin indicar el orden en que estos fueron aplicados; i.e., el propósito de este capítulo no es el de presentar implementaciones de cadenas de procesamiento, sino de presentar los bloques que pueden constituir a estas mismas. Para cada uno de estos elementos, se proporcionará una forma en que estos pueden ser implementados en MATLAB.

a. Filtros digitales. Los filtros permiten, desde una perspectiva práctica, aplicar un procesamiento sobre la señal para reducir la magnitud de una componente indeseada de la señal o, desde una perspectiva equivalente, seleccionar las componentes de interés de la señal. Dichas componentes pueden ser frecuenciales (e.g., si se desean rechazar oscilaciones que ocurren a una frecuencia superior a los 30 Hz) o, en el caso en que se tengan varias mediciones desde puntos diferentes para un mismo fenómeno, tales componentes pueden ser espaciales; e.g., si se contaran con dos señales de audio reproducidas desde diferentes ubicaciones pero dentro de un mismo ambiente y el sonido resultante es grabado desde dos localidades distintas, utilizando estas dos grabaciones podría realizarse un filtro espacial para separar el sonido proveniente de una fuente del que procede de la otra.

Cuando el filtro recibe como entrada señales digitales (i.e., señales discretas en el tiempo y en su amplitud), se dice que dicho filtro es digital. Los filtros digitales presentan la ventaja de que pueden ser elaborados dentro de un circuito o programados en una computadora, realizando operaciones básicas entre

las muestras recibidas y salidas anteriores del mismo filtro digital. Además, para el caso de este trabajo, se contarán con una serie de señales que varían no solo en el tiempo, sino también en el espacio, puesto que se cuentan con varios electrodos que están adquiriendo señales provenientes desde diferentes ubicaciones del cuero cabelludo. Cuando se realicen operaciones de filtrado utilizando únicamente la información de un mismo canal, pero diferentes muestras de este mismo, se dirá que el filtro es uno temporal, puesto que está fijando la variación en el espacio de las señales para enfocarse únicamente en las variaciones en el tiempo. En dado caso las operaciones del filtro utilicen las señales de todos los canales correspondientes a un mismo tiempo, se dirá que el filtrado es espacial, dado que está fijando el tiempo para poder considerar las variaciones presentadas entre los diferentes canales. A continuación, se presentarán distintas implementaciones de cada uno de estos filtros.

1) Filtros digitales temporales. Los filtros digitales temporales se caracterizan por operar sobre diferentes datos provenientes de una misma fuente, pero correspondientes a distintos tiempos. En dado caso un filtro tenga una respuesta impulsional de duración finita, el filtro digital es llamado FIR (de las siglas en inglés para *Finite Impulse Response*), mientras que en el caso en el cual la respuesta impulsional es de duración infinita, el filtro digital es llamado IIR (de las siglas en inglés para *Infinite Impulse Response*). Sin importar si el filtro es FIR o IIR, su salida puede expresarse con la siguiente ecuación de diferencias:

$$y[n] = \sum_{i=0}^{N} a_i x[n-i] - \sum_{i=1}^{N} b_i y[n-i]$$

Donde $y[n]$ corresponde a las secuencia de salida del filtro; $x[n]$, a la secuencia de entrada; a_i a los coeficientes de alimentación y b_i, a los coeficientes de retroalimentación [39]. Por ser un filtro digital, $x[n]$ corresponde a una lista cuyos elementos son todos correspondientes a una misma ubicación (i.e., electrodo), pero correspondientes a diferentes muestras.

Si bien ambos filtros digitales FIR e IIR comparten la forma de la ecuación característica anterior, existen diferencias entre los valores que los coeficientes de retroalimentación pueden tomar en cada uno de estas clasificaciones. En el caso de los filtros FIR, todos los coeficientes de retroalimentación b_i son iguales a cero, lo cual quiere decir que la salida puede ser calculada únicamente a partir de la muestra actual y de muestras anteriores. Por su parte, los filtros digitales IIR se caracterizan por ser al menos uno de sus coeficientes b_i distinto de cero, con lo cual se dice que el filtro presenta retroalimentación, ya que para calcular el siguiente valor en la secuencia de salida $y[n]$, además de utilizar valores actuales y previos de la secuencia de entrada, se usan también valores previos de la secuencia de salida [39].

Dentro de las ventajas de los filtros IIR frente a los filtros FIR es que, para lograr características de filtrado similares, los filtros IIR requieren de menos operaciones y memoria que los filtros FIR. Por otra parte, los filtros FIR presentan la ventaja de ser más sencillos de implementar en arquitecturas de punto fijo y de ser

inherentemente estables [41]. A continuación, se presentan los métodos que fueron utilizados para diseñar cada uno de estos filtros.

a) Diseño de filtros digitales temporales FIR. Para el diseño de los filtros digitales temporales FIR pasa-baja, se sigue el procedimiento de ventanas, descrito por Punskaya [70]:

1) Se obtiene la secuencia $h[n]$ correspondiente a la respuesta impulsional del filtro pasa-baja ideal que se desea diseñar.
2) Se obtiene $\tilde{h}[n] = h\left[n - \frac{N-1}{2}\right] \times w[n]$, donde $w[n]$ corresponde a la ventana de Hamming. Dicha ventana está definida como $w[n] = 0.54 - 0.46Cos\left[2\pi \frac{n}{N-1}\right]$ para $0 \leq n \leq N-1$ y como $w[n] = 0$ para otros valores. El resultado de dicho producto, $\tilde{h}[n]$, corresponde a la respuesta impulsional del filtro FIR deseado, el cual es de orden $N-1$. Se debe de notar que la repuesta impulsional de dicho filtro es causal, por lo que puede ser implementada.

Al momento de realizar el filtro pasa-baja, se debe de tomar en cuenta que, para un filtro pasa baja-ideal con frecuencia de corte f, la respuesta impulsional es:

$$h[n] = \frac{Sin(2\pi ft)}{\pi t}\Big|_{t=nT=n/f_s} = f_s \frac{Sin\left(\frac{2\pi f}{f_s} n\right)}{\pi n} = f_s \frac{Sin(\omega n)}{\pi n}$$

Donde f_S es la frecuencia de muestreo y ω, la frecuencia angular digital (ya normalizada). Sin embargo, si se toma dicha secuencia para obtener los coeficientes del filtro digital FIR, al momento de graficar la ganancia de dicho filtro, se obtiene que este exhibe una ganancia distinta a uno en su banda de paso, como se puede verificar en la Figura 62. Para evitar esto, se suele utilizar:

$$h^*[n] = \frac{Sin(\omega n)}{\pi n} = 2f' Sinc(\omega n)$$

en lugar de $h[n]$ [49], donde f' corresponde a la frecuencia digital normalizada. Al aplicar el procedimiento recién descrito usando $h^*[n]$ en lugar de $h[n]$, se termina teniendo un filtro con una ganancia unitaria en su banda de paso, como se ilustra en la Figura 63. En ambos casos, los filtros obtenidos presentan una fase lineal en la banda de paso, como se puede observar en la Figura 62 y en la Figura 63.

Ahora bien, si lo que se desea es implementar un filtro pasa-alta, este puede ser realizado como la resta entre dos sistemas con la misma entrada: Uno de ellos es un filtro pasa-todo y el que resta, un filtro pasa-baja, cuya frecuencia de corte es la frecuencia de paso del filtro pasa-alta deseado [35]. Esto se ilustra en la Figura 64. Para obtener la respuesta impulsional de este sistema, basta con restar de la respuesta impulsional del sistema pasa-todo (el cual corresponde a un impulso) la respuesta impulsional del sistema pasa-baja (el cual es el $\tilde{h}[n]$ calculado por el método ya presentado). Al realizar lo anterior, se debe de observar que, dado que la respuesta impulsional del filtro pasa-bajas se encuentra retrasada $\frac{N-1}{2}$ muestras para que el filtro sea

causal, la respuesta impulsional del filtro pasa-todo también debe ser retrasada; i.e., la respuesta impulsional del sistema pasa-todo será igual a 1 cuando $n = \frac{N-1}{2}$, mientras que en cualquier otra muestra, será cero.

Figura 62. Ganancia y fase de un filtro FIR diseñado utilizado $h[n]$.

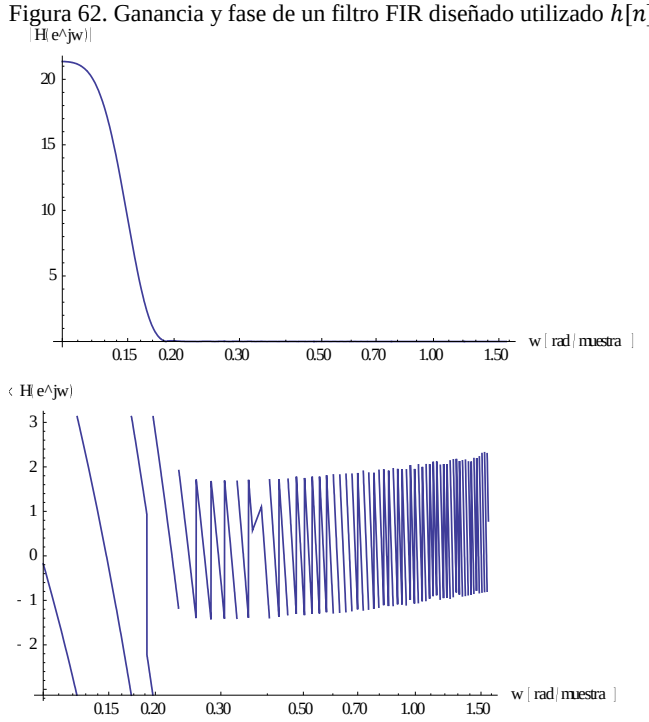


Figura 63. Ganancia y fase de un filtro FIR diseñado utilizando $h^*[n]$.

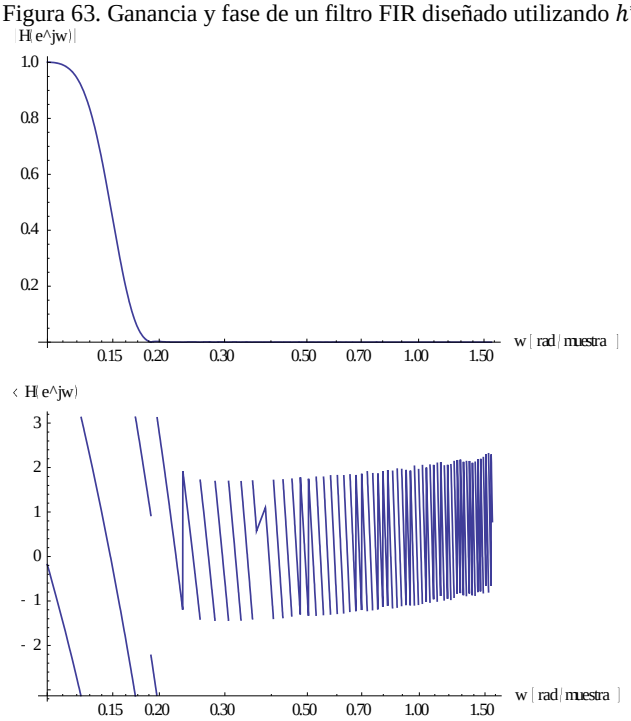
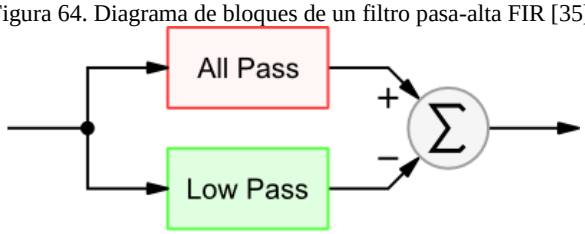


Figura 64. Diagrama de bloques de un filtro pasa-alta FIR [35].



Por otra parte, si se desea implementar un filtro pasa-banda, este podría implementarse como la conexión en serie de un filtro pasa-alta y un filtro pasa-baja. No obstante, una forma más eficiente de implementar dicho filtro es utilizar la misma estructura de la Figura 64, pero cambiando el filtro pasa-todo por otro filtro pasa-baja [35]. De esta forma, ambos filtros recibirán la señal de entrada y el filtro colocado en la parte superior de la Figura 64 dejará pasar las frecuencias digitales menores que ω_H, mientras que el filtro colocado en la parte inferior dejará pasar las frecuencias digitales menores que ω_L. No obstante, luego se restarán las respuestas del filtro, por lo que a una señal que contiene todas las frecuencias menores que ω_H se le restará una señal que contiene todas las frecuencias menores a ω_L, resultando así una señal que únicamente contiene frecuencias entre ω_L y ω_H.

El segundo método es equivalente al primero, como lo puede mostrar un análisis de diagramas de bloques: Si se inicia con el filtro pasa-banda conformado por un filtro pasa-baja y uno pasa-alta interconectados en serie, se puede iniciar ingresando el bloque del filtro pasa-baja dentro de los dos bloques que conforman al filtro pasa-alta. Así, se obtendrá al filtro pasa-baja en serie con el filtro pasa-todo, lo cual equivale al mismo filtro pasa-baja, y también se tendrá al filtro pasa-baja en serie con el filtro pasa-baja que conforma al filtro pasa-alta, lo cual será equivalente al filtro pasa-baja con la menor frecuencia de corte. Para que el filtro pasa-banda presente una banda de paso, este último filtro pasa-baja con la menor frecuencia de corte debería de corresponder al filtro pasa-baja que conformaba al filtro pasa-alta. Al mismo tiempo, este método para el diseño de filtros pasa-banda es más eficiente en su implementación que el filtro pasa-banda visto como dos sistemas en serie, ya que para obtener la respuesta impulsional en este caso basta con realizar una resta de respuestas impulsionales, mientras que en el otro caso, tendría que realizarse una resta entre respuestas impulsionales para obtener la respuesta del filtro pasa-alta y luego, convolucionar este resultado con la del filtro pasa-baja, con el fin de obtener la respuesta impulsional del filtro pasa-banda.

Con el objetivo de poder diseñar rápidamente filtros pasa-banda FIR, se realizó una función en MATLAB para generar la respuesta impulsional de un filtro deseado, la cual se muestra a continuación. Debe de notarse que dicha repuesta impulsional corresponde a los coeficientes de alimentación a_i que multiplicarán a la muestra actual y pasadas en la ecuación de diferencias para obtener la salida actual del filtro.

```
function coef = coefFiltroFIR(fl,fh,fs,N)
%fl corresponde a la frecuencia de paso del filtro pasa-banda;
%fh, a la frecuencia de corte (fh>fl); fs, a la frecuencia de muestreo
%y N, al número de muestras a tomar para la ventana.

fth = fh/fs;         %Frecuencia de corte normalizada
ftl  = fl/fs;        %Frecuencia de paso normalizada
coef = zeros(N,1);  %Lista de coeficientes de alimentación del filtro.

for n = 0 : N-1
    coef (n+1, 1) = (2 * fth * sinc( 2*fth * (n-(N-1)/2)) - 2 * ftl …
                     * sinc( 2*ftl* (n-(N-1)/2)))*…
                     (0.54 - 0.46*cos( 2*pi*n/(N-1) ) );
end
```

Donde las funciones seno cardinal (*sinc*) no contienen al término π, puesto que MATLAB ya lo incluye dentro de su implementación de la función.

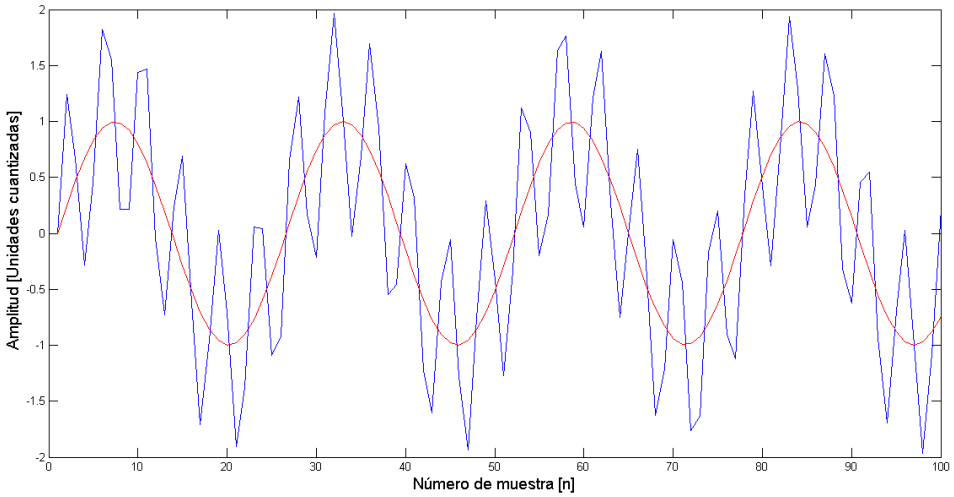
Para aplicar el filtro obtenido con el método anterior sobre una secuencia $x[n]$, podría haberse realizado un producto punto entre el vector *coef* obtenido como salida de la función anterior y un vector que contuviera N muestras consecutivas de $x[n]$, donde dichas muestras se irían corriendo de una en una, como en una ventana móvil. De hacer esto, el filtro habría introducido un retraso en la señal de salida, debido a su fase lineal.

No obstante, existe una técnica para evitar dicho retraso, la cual es conocida como un filtro de fase cero. La aplicación de dicho filtro requiere del conocimiento de muestras posteriores a la actual, por lo que no se puede realizar en una aplicación de tiempo real. Pero si se está realizando una aplicación el procesamiento de la señal fuera de línea, esta opción constituye una alternativa válida. Una de las maneras en que un filtro de orden cero puede ser aplicado consiste en que primero, se aplica el filtro como normalmente se haría, sobre la secuencia $x[n]$, obteniendo a la salida una secuencia $x'[n]$. Luego, se obtiene una versión inversa en el tiempo de la señal de salida, i.e., $x'[-n]$ y a dicha secuencia, se le vuelve a aplicar el mismo filtro. El resultado será una secuencia $y[n]$ cuya fase en la banda de paso es de cero [83]. Además, este filtro de fase cero equivale a un filtro con una ganancia que es igual al cuadrado de la ganancia del filtro aplicado dos veces en su implementación [83]. Para poder aplicar un filtro FIR de fase cero en MATLAB, basta con utilizar el siguiente comando:

```
y = filtfilt(coef,1,x);
```

Donde *coef* corresponde a la lista de coeficientes del filtro FIR y *x*, al vector que contiene la secuencia a filtrar. En la Figura 65, se muestra un ejemplo en el cual se aplica un filtro FIR pasa-banda de 1 a 15 Hz sobre la señal de entrada $x(t) = \sin(2\pi * 5 * t) + \sin(2\pi * 30 * t)$, muestreada a 128 Hz. Se puede observar que la señal de salida corresponde prácticamente, sin distorsión ni desfase, a $\sin(2\pi * 5 * t)$, lo cual se logró al utilizar un filtro de fase cero y orden elevado de 255.

Figura 65. Señal de entrada, en azul, y de salida, en rojo, para un filtro FIR de fase cero y orden 255.



Con la función *coefFiltroFIR* realizada y el comando *filtfilt* de MATLAB, se tenían ya medios para realizar un filtrado casi ideal de la señal, puesto que al utilizar un filtro de orden cero no se introduce desfase y, además, al usar un N grande, se toman más muestras de la respuesta impulsional, con lo cual el filtro implementado se aproxima más al comportamiento de uno ideal. Todo esto viene con el costo de una mayor cantidad de operaciones, puesto que se debe de aplicar dos veces un filtro de orden considerablemente grande, además de que este filtro no puede ser utilizado en aplicaciones de tiempo real, por requerir el conocimiento de muestras correspondientes a tiempos futuros.

b) Diseño de filtros digitales temporales IIR. Para el diseño de los filtros digitales IIR, se utilizó el método dado por Oppenheim, Schafer y Buck [64], en el cual se inicia con un prototipo pasa-bajas y se utiliza la transformación bilineal para relacionar el dominio s con el dominio z. El procedimiento se muestra a continuación, para el diseño de un filtro pasa-banda:

1) Se mapean las frecuencias digitales de corte y de paso deseadas a frecuencias analógicas, utilizando la siguiente expresión:

$$\Omega = \frac{2}{T}\tan\left(\frac{\omega}{2}\right)$$

Donde ω corresponde a una frecuencia digital; T, al período de muestreo y Ω, a la frecuencia analógica correspondiente a ω. Puesto que se desea realizar un filtro pasa-banda, al finalizar este paso se debe de tener una frecuencia analógica de paso, Ω_L, y una de corte, Ω_U, donde $\Omega_U > \Omega_L$. Ω_U corresponde al mapeo de la frecuencia digital ω_U y Ω_L, al de ω_L.

2) Sobre el prototipo de un filtro Butterworth pasa-baja de segundo orden, cuya función de transferencia viene dada por:

$$H(s) = \frac{1}{s^2 + \sqrt{2}s + 1}$$

Se realiza la sustitución:

$$s = \frac{s^2 + \Omega_L \Omega_U}{(\Omega_U - \Omega_L)s}$$

Obteniendo una nueva expresión para $H(s)$ que corresponde un filtro pasa-banda con frecuencia de paso Ω_L y de corte Ω_U.

3) Se aplica la transformación bilineal sobre la función de transferencia obtenida en el paso anterior. Para ello, se realiza la sustitución:

$$s = \frac{2}{T}\frac{z-1}{z+1}$$

Debido a que el procedimiento para este diseño de filtros requiere la manipulación algebraica de varias expresiones matemáticas, se realizó un programa en *Mathematica* que permitía diseñar los filtros IIR siguiente los pasos descritos anteriormente. Dicho programa se presenta a continuación:

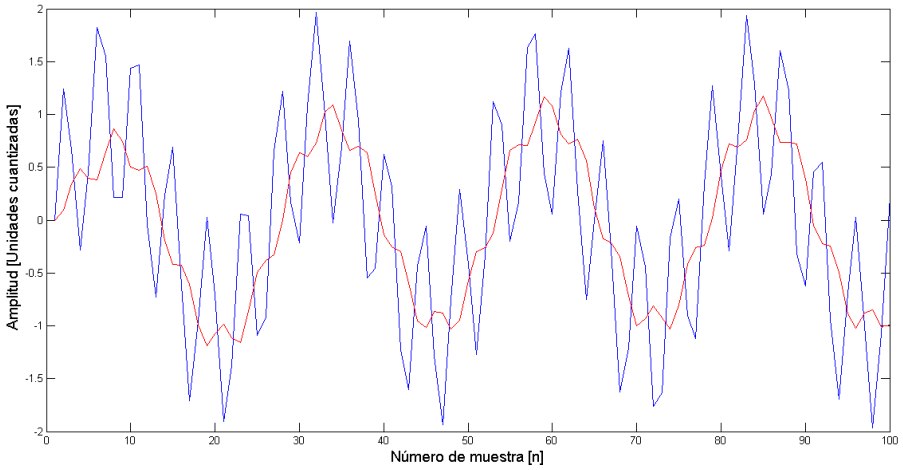
```
wl = 2*Pi;    (*Frecuencia angular de paso del filtro pasa-banda.*)
              (*En este caso, es 1 Hz.*)
wu = 20*Pi;   (*Frecuencia angular de corte del filtro pasa-banda.*)
              (*En este caso, es 10 Hz.*)
fs = 128;     (*Frecuencia de muestreo.*)
ts = 1/fs;    (*Período de muestreo.*)

(*Frecuencia analógica correspondiente a la frecuencia digital baja*)
Ol = 2/ts *Tan[wl*ts/2];
(*Frecuencia analógica correspondiente a la frecuencia digital alta*)
Ou = 2/ts *Tan[wu*ts/2];

(*Filtro Butterworth pasa-bajas prototipo*)
Prototipo[s_] = 1/(s^2+Sqrt[2]*s+1);
(*Filtro Butterworth pasa-banda analógico*)
PrototipoPB[s_] = Prototipo[(s^2+Ol*Ou)]/((Ou-Ol)*s);
(*Filtro deseado*)
Filtro[z_] = Simplify[ N[ PrototipoPB[ 2/ts * (z-1)/(z+1) ] ] ];
```

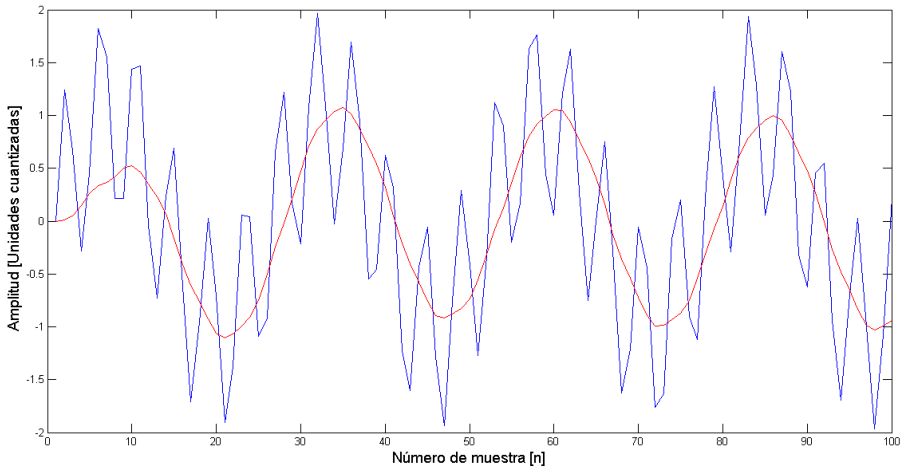
Una vez obtenida la función de transferencia del filtro deseado, *Filtro[z]*, utilizando *Mathematica*, dicha función era re-expresada como una ecuación de diferencias, la cual era implementada en MATLAB, utilizando condiciones iniciales iguales a cero. Para verificar la correcta operación de los programas mencionados, se diseñó un filtro IIR Butterworth, de segundo orden, pasa-banda para el rango de frecuencias entre 1 y 15 Hz. Dicho filtro fue alimentado con la señal $x(t) = \sin(2\pi * 5 * t) + \sin(2\pi * 30 * t)$, muestreada a 128 Hz, y, tanto dicha señal como la salida del filtro, se ilustran en la Figura 66. En dicha figura, se puede apreciar como el filtro atenúa significativamente a la componente de 30 Hz, mientras conserva la de 5 Hz.

Figura 66. Señal de entrada, en azul, y de salida, en rojo, para un filtro IIR de segundo orden.



En dado caso se quería tener un filtro de orden mayor, se conectaban varios de estos filtros en serie. La implementación de dicha conexión se realizó como una aplicación de la ecuación de diferencias sobre la secuencia obtenida a la salida del filtro anterior; i.e., no se implementó con una sola ecuación de diferencias, sino como varias en cascada. La ventaja de este método es que se podía modificar con facilidad el orden del filtro utilizado al seleccionar la salida correspondiente a la etapa de interés. Al aumentar el orden del filtro, se mejoran sus características frecuenciales, principalmente a través de la reducción de las bandas de transición, la cual va acompañada de un aumento en el valor absoluto de la pendiente de la ganancia en dichas bandas. Como muestra de lo anterior, en la Figura 67, se muestra la respuesta para un filtro que recibe la misma entrada y tiene las mismas características del filtro de la Figura 66, con la única diferencia de que ahora el filtro es de cuarto orden (i.e., está conformado por dos filtros como los de la Figura 66 en serie). Se puede observar que la componente de frecuencia de 30 Hz fue prácticamente removida y, en el estado estable, la señal de salida del filtro corresponde a una señal sinusoidal de 5 Hz.

Figura 67. Señal de entrada, en azul, y de salida, en rojo, para un filtro IIR de cuarto orden.



c) Comparación y uso del diseño de filtros FIR e IIR. Como se pudo observar a lo largo de la presentación de los filtros FIR e IIR, los filtros FIR presentan la ventaja de poder ser diseñados de una manera más sencilla que no requiere de muchas manipulaciones algebraicas y que puede ser directamente automatizado dentro de MATLAB a través de unas funciones de programación. Esto permitía un rápido diseño de filtros que solo requería del cambio de parámetros en la función *coefFiltroFIR*. Otra ventaja de los filtros FIR es que podían implementarse como funciones de fase cero y además, darles un orden lo suficientemente grande como para que sus características fuera casi ideales. De esta forma, el filtro permitiría la recuperación precisa de un rango de frecuencias de interés. No obstante, estas ventajas no son aplicables a un procesamiento en tiempo real.

Por su parte, los filtros IIR presentaban un mayor reto al momento de su diseño, puesto que, a pesar de que se obtenían sus funciones de transferencia en el dominio Z utilizando *Mathematica*, dichas funciones luego debían de ser expresadas como ecuaciones de diferencias, las cuales se codificaron directamente a mano (i.e., sin un proceso automático). De tal manera que, cada vez que se deseaba cambiar algún parámetro del filtro, incluyendo su orden, debían de realizarse modificaciones en el código correspondiente. Sin embargo, los filtros IIR requerían de un número significativamente reducido de operaciones para obtener resultados similares a los de los filtros FIR; e.g., al comparar la Figura 65 y Figura 67, se puede apreciar que las salidas de ambos filtros no son tan distintas, a pesar de que uno de ellos es FIR de orden 255 y el otro, IIR de orden 4.

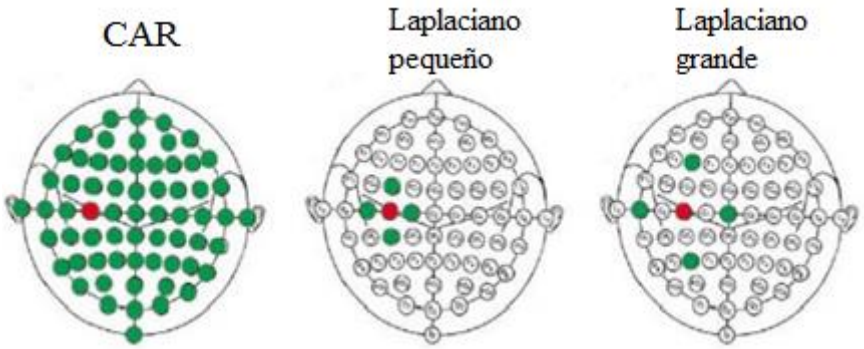
Dadas las ventajas y desventajas de ambos tipos de filtros, estos fueron utilizados de la siguiente manera: Para las primeras pruebas en las cuales no se conocía con precisión la banda de frecuencia a filtrar, se utilizó el filtro FIR sobre datos ya grabados. Así, probando y contrastando diferentes opciones, se determinó la banda de frecuencias que se deseaba filtrar. Ya con dicha información, se creaba un filtro IIR con las frecuencias de paso y de corte determinadas, el cual era probado de nuevo sobre los mismos datos grabados para verificar su efectividad y luego, dicho filtro era agregado al programa de procesamiento en tiempo real. La métrica utilizada para determinar cuáles características de filtrado eran las mejores fue el puntaje F_1 obtenido al finalizar el entrenamiento del algoritmo clasificador.

Por otra parte, la razón por la cual se diseñaron únicamente filtros pasa-banda y no de otro tipo es la siguiente: Dentro de las señales de EEG, existen componentes frecuenciales menores y mayores a las de la onda P300, por lo que se necesita filtrar tanto frecuencias altas como bajas. Asimismo, cuando el paciente realiza algún movimiento (e.g., de su cabeza o de sus ojos), dichos movimientos producen señales eléctricas que son captadas por los electrodos de EEG. No obstante, dichas señales generadas son de baja frecuencia, por lo que pueden ser retiradas utilizando un filtro pasa-alta.

2) Filtros digitales espaciales. Al trabajar con señales EEG que fueron grabadas utilizando más de un electrodo, se pueden aplicar filtros espaciales, dado que se tendrá varias muestras para un mismo tiempo, pero en diferentes ubicaciones. Idealmente, estos filtros tienen por objetivo el aislar las fuentes de campos eléctricos en el cerebro, puesto que las señales eléctricas generadas por cada una de estas fuentes se suman con las señales provenientes de otras fuente; i.e., las señales obtenidas en cada electrodo no corresponden exclusivamente a la actividad eléctrica cerebral en la región donde se encuentra el electrodo, sino que dicha señal también presenta componentes, si bien atenuadas, de la actividad cerebral en otras regiones de la cabeza. Estos filtros también son útiles para reducir artefactos tales como movimientos de cabeza, ya que generalmente estos movimientos inducen un potencial que afecta a varios electrodos.

Dentro de los filtros digitales espaciales más utilizados para mejorar la relación de señal a ruido de señales EEG, se encuentran la referencia del promedio común (también conocida como CAR, por las siglas en inglés de *Common Average Reference*) y los Laplacianos de superficie, los cuales pueden ser un Laplaciano grande o un Laplaciano pequeño [45]. Estos filtros digitales se ilustran en la Figura 68 y se explicarán a continuación.

Figura 68. Filtros espaciales más utilizados con señales de EEG.



a) Referencia del promedio común (CAR). En este filtro se toman, para un mismo tiempo, todas las muestras provenientes de los diferentes electrodos y se obtiene un promedio de dichas muestras. El resultado es restado de cada uno de las muestras de los canales [45]; i.e., cada una de las muestras estará referenciada con respecto al promedio de todas las señales en un momento específico. Si *signal* es una matriz que tiene en sus filas los diferentes canales de EEG y en sus columnas, las muestras de cada canal a través del tiempo, el CAR puede implementarse en MATLAB con la siguiente línea de código:

```
signalCAR = bsxfun(@minus,signal,mean(signal));
```

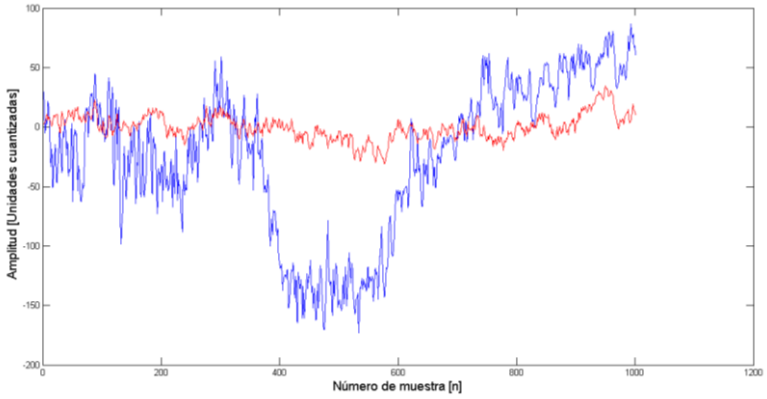
b) Laplaciano de superficie. En los Laplacianos, a diferencia de con CAR, solamente se utilizan los electrodos cercanos al electrodo de interés para obtener una nueva referencia. Además, en el cálculo de dicha referencia, no todos los canales tienen el mismo peso en el promedio, sino que se utiliza un promedio ponderado en el cual los pesos son funciones de la distancia entre el electrodo a filtrar y cada uno de los

electrodos a considerar. Si la distancia es corta (i.e., se utilizan los electrodos más cercanos al de interés) se dice que el Laplaciano es pequeño; en caso contrario, se dice que el Laplaciano es grande.

c) Comparación de los filtros digitales espaciales. Se han realizado estudios en los cuales se ha determinado que, dentro de las técnicas presentadas anteriormente, aquellas que proporcionan una mayor mejoría de la relación de señal a ruido son el CAR y el Laplaciano pequeño. Dado que el CAR es más sencillo de implementar, no solo por la cantidad de operaciones a realizar, sino también porque no requiere de una calibración de los pesos de cada uno de los canales, únicamente se utilizó este filtro espacial al momento de probar diferentes cadenas de procesamiento; i.e., se eligió CAR por ofrecer una mejora de señal a ruido comparable con la del Laplaciano pequeño, al mismo tiempo de ser una opción que requiere de una menor cantidad de operaciones y que no necesita una calibración única para cada sujeto con el cual se utilice (Ya que, si bien los electrodos del Emotiv EPOC están guiados por su estructura, dicha estructura es flexible y las posiciones finales de los electrodos pueden variar según la forma de la cabeza del usuario).

Para verificar y dar un ejemplo de la utilidad del filtro espacial CAR, se muestra la Figura 69. En ella, se ilustra en azul la señal de uno de los electrodos de EEG antes de aplicársele CAR y luego de, en color rojo. Se puede observar que, si bien la señal disminuyó en amplitud, se retiró el ruido común a todos los electrodos, el cual se hallaba deformando la señal. Muestra de esto último se presenta principalmente entre la muestra 400 y 600, donde la señal sin CAR exhibe una oscilación negativa, la cual pudo haber sido causada por algún movimiento del paciente.

Figura 69. Señal de entrada, en azul, y de salida, en rojo, para un filtro espacial CAR.



b. Normalización de las señales de EEG. La normalización es un paso comúnmente utilizado al trabajar con algoritmos de aprendizaje automático, ya que permiten escalar las características a considerar a un rango definido, evitan problemas numéricos durante la operación del clasificador [38], además de que facilitan la convergencia de dichos algoritmos [61]. Si se tiene un conjunto de datos X, con una media

aritmética μ y una desviación estándar σ, una de las técnicas de normalización más utilizadas es obtener la unidad tipificada z [61], para una medición x_i, de la siguiente manera:

$$z = \frac{x_i - \mu}{\sigma}$$

Para este trabajo, dependiendo de las características comunes al conjunto de datos X, la normalización recibirá diferentes nombres. Será llamada espacial si todos los datos en X considerados para calcular el promedio y la desviación estándar corresponden a muestras provenientes de diferentes canales, pero correspondientes al mismo tiempo. En dado caso los datos provengan todos de un mismo canal, pero para diferentes tiempos, se dirá que la normalización es temporal. Si *signal* es una matriz que tiene en sus filas los diferentes canales de EEG y en sus columnas, las muestras de cada canal a través del tiempo, la normalización espacial se realizaría, en MATLAB, de la siguiente manera:

```
mu = mean(signal);
desv = std(signal);
signalNorm = bsxfun(@minus,signal,mu);
signalNorm = bsxfun(@rdivide,signalNorm,desv);
```

El resultado normalizado estaría en la matriz *signalNorm*. Por otra parte, la normalización temporal sería:

```
mu = mean(signal')';
desv = std(signal')';
signalNorm = bsxfun(@minus,signal,mu);
signalNorm = bsxfun(@rdivide,signalNorm,desv);
```

La normalización también puede no ser temporal ni espacial, sino por características: En dicho caso, el conjunto *X* está conformado por datos provenientes de diferentes ejemplos, pero que corresponden a la misma característica en todos ellos; e.g., en esta aplicación, todas las primeras muestras del primer canal corresponden a una misma característica.

De igual forma, si la normalización fue realizada tomando en cuenta únicamente datos provenientes de un solo intervalo luego del estímulo mostrado por el teclado P300, se dirá que la normalización fue por estímulo. Si, antes de normalizar, se obtiene la respuesta promedio para una misma letra y luego, se normaliza, se dirá que la normalización fue por letra. Si se realizó lo anterior, pero tomando en cuenta no solo los datos de una letra, sino de toda una corrida, se dirá que la normalización fue realizada por sesión. Por otra parte, en el caso en que se consideró toda la información de una sola corrida, previo a que esta fuera separada por estímulos o promediada, se dirá que la normalización fue por corrida. Y, si los datos considerados corresponden a todos los recopilados durante las sesiones de entrenamiento, se dirá que la normalización fue total.

c. Promedio de las señales de EEG posteriores al estímulo. En el capítulo de adquisición de las señales EEG, se mostró cómo se creaban listas en las cuales se almacenaban cada uno de los segmentos de interés posteriores a los estímulos, incluyendo su clasificación como estímulo objetivo o no objetivo, y el código de la fila o columna a la cual tal estímulo correspondía. Con dicha información, puede realizarse fácilmente una selección de los estímulos correspondientes a un mismo elemento de la matriz y promediarlos para mejorar la relación de señal a ruido, como ya se explicó en el marco teórico.

d. Decimación. La decimación cosiste en, una vez muestreada una señal banda-limitada, simular su re-muestreo a una frecuencia menor a la original [40]. Para lograr esto, se deben de realizar dos pasos: El primero de ellos es aplicar un filtro pasa-baja digital que banda-limite a la señal muestreada a una frecuencia menor a la mitad de la nueva frecuencia de muestreo. Luego de esto, se seleccionan muestras separadas por la misma cantidad de muestras, de tal forma que se tome una muestra de cada M, donde M corresponde al factor de decimación.

Para ilustrar el anterior concepto, se utilizará un ejemplo: Supóngase que se tiene una señal banda-limitada a 64 Hz, la cual fue muestreada a 128 Hz. No obstante, ahora se desea decimar la señal por un factor de 4. Esto quiere decir que se desean tener las muestras que se habrían obtenido de haberse hecho un muestreo a 32 Hz. Para lograr esto, primero se debe de aplicar un filtro digital que tenga una frecuencia de corte de, como máximo, 16 Hz. Una vez hecho lo anterior, se seleccionan las muestras de la señal de salida cuyo número de muestra son múltiplos de 4. De esta forma, por cada segundo de muestreo, se pasará de tener 128 a 32 muestras.

La decimación es un método de procesamiento de señales que permite reducir la dimensión de los datos, lo cual es de suma utilidad al trabajar con algoritmos de aprendizaje automático, los cuales son sensibles a la maldición de la dimensión.

3. Clasificación de las señales de EEG. En los capítulos anteriores, se ha tratado el tema de la adquisición y procesamiento de las señales EEG. Pero estas dos tareas han sido realizadas con un solo y mismo objetivo: El poder realizar una adecuada clasificación de la señal para determinar la letra en la cual el usuario de un teclado P300 estaba concentrado. En este capítulo, se presentará el algoritmo de aprendizaje automático seleccionado, así como la implementación usada en este trabajo. Además, se mostrarán los criterios y métodos utilizados para sintonizar los parámetros de dicho algoritmo y reportar su rendimiento.

a. Selección del algoritmo de aprendizaje automático. En el marco teórico del presente trabajo, se presentaron distintos algoritmos de aprendizaje automático, incluyendo las ventajas, desventajas y desempeño de estos al ser utilizados en aplicaciones de P300. Los estudios realizados por Krusienski [44] y Manyakov [48] concluyeron que uno de los mejores clasificadores para trabajar con la señal de P300 eran los SVM. Si bien estos eran superados en ambos estudios por diferentes versiones de análisis de discriminante

lineal, los SVM presentaban también la ventaja de requerir pocos datos para su entrenamiento [44], así como de ser insensibles a la maldición de la dimensión [46]. Estas dos ventajas hacen del SVM un clasificador muy atractivo: Primero, las sesiones de entrenamiento para el teclado P300 en las cuales se adquieren señales de EEG para entrenar al clasificador toman una cantidad de tiempo considerable (alrededor de 20 minutos cada una) y pueden fatigar rápidamente al usuario por lo que, entre menos sesiones de estas se necesiten, será mejor. Segundo, el hecho de que puedan tomarse en cuenta un conjunto más amplio de características al momento de entrenar al clasificador puede ser útil para hacer del sistema uno que pueda ser utilizado con varias personas: De esta forma ya no se deben de seleccionar características específicas de las señales según una persona (las cuales podrían variar entre personas), sino que el clasificador es capaz de recibir una entrada de mayor dimensión sin requerir de más datos de ejemplo.

Dentro de las desventajas que Krusienski [44] presentaba con respecto a los SVM, se hallaba que su algoritmo es complicado y que toma un poco más de tiempo de entrenamiento. Sin embargo, para este trabajo, dichas desventajas no son un factor determinante para no adoptar el SVM, puesto que la implementación del algoritmo será llevada a cabo en una computadora, la cual cuenta con recursos de memoria y procesamiento suficientes para ejecutarlo. Además, debido a que el entrenamiento del clasificador no ocurre en tiempo real, es de poca importancia si este mismo toma un par de minutos (como se observó que duraban los entrenamientos del SVM). En cuanto a la predicción de la letra que sí debe de ser realizada en tiempo real, el tiempo que toma este algoritmo en realizar dicha predicción es corto en escalas de tiempo sensibles (i.e., menor a 1 s). Por tanto, si bien dicha clasificación podría tomar un tiempo significativo en la escala de procesamiento del programa (i.e., en el orden de los μs), dicho tiempo no será perceptible para el usuario y puede realizarse al finalizar la captura de señales de EEG, para no interferir con ninguna de las tareas de adquisición y procesamiento de la señal.

Krusienski [44] también mencionaba que dentro de las desventajas del SVM, específicamente del SVM de *kernel* gaussiano, se hallaba el hecho de que este podía fácilmente sobre-ajustarse a los datos de entrenamiento dados. Al mismo tiempo, mencionaba que quizás, con un esfuerzo adicional, el clasificador podía ajustarse para presentar un mejor rendimiento. De la teoría se sabe que los SVM cuentan con un parámetro de regularización *C*, el cual fue específicamente diseñado para evitar el sobre-ajuste del clasificador. En el caso específico de los SVM de *kernel* gaussiano, estos también cuentan con el parámetro σ, el cual también juega un rol en evitar que el clasificador se sobre-ajuste. Como un contraejemplo más a lo presentado por Krusienski [44], se tienen los resultados del trabajo de Manyakov [48], en el cual el SVM de *kernel* gaussiano se posiciona como el segundo mejor dentro de los comparados, lo cual no podría haberse logrado si dicho clasificador se sobre-ajustara.

Ahora bien, en cuanto al *kernel* del algoritmo, se eligió el gaussiano puesto que sus características de no-linealidad pueden ayudarlos a capturar variaciones no lineales en las señales de EEG que no podrían ser

captadas por el SVM de *kernel* lineal (ni por los algoritmos de análisis por discriminante lineal). Esta decisión se ve respaldada por los resultados de Manyakov [48], en los cuales el SVM de *kernel* gaussiano exhibió un mejor desempeño que el SVM de *kernel* lineal. Dadas las razones anteriores, se eligió al SVM de *kernel* gaussiano como el algoritmo clasificador a usar.

Si bien MATLAB cuenta con una implementación propia de un SVM, Ng [61] recomienda utilizar la librería LIBSVM, la cual es de distribución abierta y presenta la ventaja de ser una versión optimizada del algoritmo de SVM [15]. Esta librería puede descargarse como un archivo .zip del sitio web https://www.csie.ntu.edu.tw/~cjlin/libsvm/. Una vez descargada, se descomprime dicho archivo y se dirige a la carpeta *windows*. Esta contiene los archivos compilados para computadoras *Windows* de 64-bits. Para una computadora *Windows* de 32-bits, los archivos deben de ser compilados utilizando la función *make* del folder *matlab*. Una vez se tengan los archivos compilados para *svmtrain* y *svmpredict*, estos deben de ser incluidos dentro de la carpeta con el código que los utilizará o en la ruta de búsqueda de MATLAB para poder ser utilizados.

La librería LIBSVM implementa diferentes tipos de SVM, no solo variando el *kernel* de estos, sino también el tipo de SVM. El SVM presentado en el marco teórico de este trabajo corresponde a un SVM tipo C y es el que usaremos para realizar las clasificaciones de las señales de EEG. Otra ventaja que presenta esta librería es que es capaz de realizar clasificaciones multi-clase (i.e., donde hay más de dos categorías para clasificar) y que puede generar a su salida estimados de la probabilidad de que un elemento clasificado pertenezca a una clase o a la otra [15].

b. Creación del modelo de clasificación. Una vez finalizada la adquisición y procesamiento de las señales EEG, se tendrá una matriz X y un vector columna y. Las filas de X corresponden a las muestras, en el dominio del tiempo y durante un tiempo posterior a un estímulo, ya procesadas de las señales de EEG. Debido a que se cuentan con múltiples electrodos de EEG, los cuales son devueltos por los medios de adquisición como una matriz, se deben de seleccionar las muestras deseadas y luego agruparlas como un vector para poderlos añadir a la matriz X, teniendo el cuidado de siempre conservar el mismo orden de los canales. Por su parte, cada una de las filas del vector y corresponde a la clasificación del elemento cuyas características se presentan en la misma fila de la matriz X. Para utilizar correctamente la librería LIBSVM, los elementos del vector y solo puede tomar uno de dos valores: 1 o -1. En este caso, se utilizará el número 1 para identificar a las señales que contienen la onda P300 como reacción al estímulo correspondiente y el número -1 para las que no.

Para entrenar el SVM implementando en SVM, basta con ejecutar los siguientes comandos en MATLAB:

```
svmoptions = strcat( {'-s 0 -t 2 -g '}, num2str(gamma),{' -c
'},num2str(C),{' -b 1 -q'} );
svmoptions = char( svmoptions );
model = svmtrain(yTrain, XTrain, svmoptions);
```

Donde C es una variable numérica igual al valor del parámetro *C* del SVM; *gamma*, es igual a $1/\sigma$; XT es una matriz que contiene muestras de la matriz *X* usadas para entrenar al algoritmo y *yT*, el vector que contiene las clasificaciones correspondientes a las filas de *XT*. Además, *svmoptions* es una cadena de texto que indica la configuración del SVM: "-s 0" indica que se utilizará un clasificador SVM tipo C; "-t 2", que dicho clasificador será de *kernel* gaussiano; "-b 1", que el clasificador será entrenado para generar probabilidades como salidas y "-q", que el algoritmo no debería de imprimir nada en consola mientras se ejecuta.

Una vez finalizado el entrenamiento del algoritmo clasificador, este devuelve un modelo de clasificación, el cual es almacenado en la variable *model*. Dicho modelo debe ser guardado para, posteriormente, realizar predicciones, con el siguiente comando:

```
[y, acc, p] = svmpredict(yTest, XTest, '-b 1 -q');
```

Donde *XTest* es la matriz con los elementos a clasificar; *yTest*, el vector con las clasificaciones de las filas de *XTest*; *y*, el vector con las clasificaciones predichas por el algoritmo; *p*, el vector con las probabilidades de que cada fila de *XTest* contenga una señal P300 en respuesta al estímulo correspondiente y *acc*, para comparar el porcentaje de elementos que son iguales entre *y* y *yTest*. La lista *yTest* es opcional y podría no incluirse. Además, al final de la instrucción se añaden los comando '-b 1 -q', indicando que la predicción debe de incluir probabilidades y que no debe de imprimir información en pantalla.

c. Validación del modelo de clasificación. Hasta ahora, ya se conoce cómo realizar clasificaciones y predicciones con el algoritmo SVM implementado en LIBSVM. Sin embargo, aún hace falta validar dicho modelo para verificar que este no se sobre-ajuste al conjunto de entrenamiento utilizado. Para ello, se inicia dividiendo la matriz X y el vector y en tres conjuntos: uno de entrenamiento, uno de validación cruzada y uno de prueba. A pesar que algunos autores, como Ng [61], recomiendan el uso de 60% de los datos para el conjunto de entrenamiento; 20%, para el de validación cruzada y 20% para el de prueba, en este trabajo se utilizó 70% para el entrenamiento; 15% para la validación y 15% para el conjunto de prueba, debido a que se deseaba contar con una mayor cantidad de datos de entrenamiento. Cada uno de estos conjuntos guardó la proporción de ejemplos de respuesta a estímulos positivos (i.e., cuando se iluminaba la letra de interés) a negativos, con el objetivo de entrenar al clasificador con datos similares a los que recibiría al operar en tiempo real. Tal proporción es de cinco estímulos negativos por cada estímulo positivo. Por lo demás, la selección de cada uno de los elementos que conformaban a cada uno de los conjuntos fue realizada al azar.

Luego, se procedió a entrenar al algoritmo clasificador fijando un valor de C y de σ, utilizando los datos correspondientes al conjunto de entrenamiento. Utilizando el modelo obtenido, se realizaba una predicción sobre el conjunto de validación y se calculaba el puntaje F_1 de dicha clasificación. El procedimiento anterior se repetía, dejando fijo el valor de C, pero variando el valor de σ dentro de una lista previamente definida. Luego, se variaba el valor de C y se volvían a barrer los valores de σ, y así hasta probar todas las combinaciones de valores de C y σ dados por las listas predefinidas. Durante todo este procedimiento, si alguno de los modelos presentaba un mayor puntaje F_1 que el obtenido hasta ahora, se almacenaban los parámetros (C y σ) y el modelo correspondiente. Los valores de C fueron barridos dentro de los valores de una serie geométrica que iniciaba en 1×10^{-4} y finalizaba en 30, siendo tres la razón de la serie. Por su parte los valores de σ fueron barridos dentro de una serie geométrica que iniciaba en 0.1 y finalizaba en 3×10^{5}, siendo tres la razón de la serie. Los límites de esta serie fueron determinados de forma experimental, mientras que su razón fue tomada de Ng [61].

Al finalizar el procedimiento anterior, se tendría un modelo que no se sobre-ajustaba a los datos de entrenamiento, puesto que se había verificado que este ofrecía el mayor puntaje F_1 sobre un conjunto de datos que el clasificador no había utilizado durante su entrenamiento. Para reportar una métrica del rendimiento de dicho modelo, este era utilizado para realizar una predicción sobre el conjunto de prueba. El puntaje F_1 obtenido en dicha predicción era utilizado como la métrica de desempeño del clasificador.

Se utilizó el puntaje F_1 y no otra métrica de desempeño por las siguientes razones: Como primer punto, los datos que se estaban clasificando eran datos binarios sesgados, donde una de las clases sucedía con una frecuencia cinco veces mayor a la de la otra. Por tanto, utilizar la exactitud como métrica podría causar confusiones, ya que un clasificador que siempre predijera -1 tendría una exactitud del 83.33%. Por otra parte, considerando una clasificación puramente binaria, en esta aplicación no nos interesa tener mucha exhaustividad y poca precisión, puesto que en dicho caso, sería difícil decidir cuál de los elementos clasificados como positivos corresponde a los de la letra correcta; así como no nos interesa tener una gran precisión y una baja exhaustividad, ya que podrían haber casos donde solo podamos predecir con predicción la columna o fila y no el otro elemento. En cambio, nos interesa tener una buena precisión y exhaustividad, ya que nos interesa tener aquellos elementos que, con menos incertidumbre, se asemejan a la clasificación positiva. Y, como ya se presentó en el marco teórico, la métrica para tomar en cuenta estos dos aspectos es el puntaje F_1.

4. Cadenas de procesamiento implementadas. Ya conociendo los elementos que permitían la adquisición, procesamiento y clasificación de las señales, así como las maneras de implementarlos para poder realizar un sistema funcional, se procedió a realizar diferentes cadenas de procesamiento y a comparar los resultados brindados por ellas. Para poder comparar dichos resultados, se grabó, para cada sujeto, un conjunto de datos adicional, el cual no fue utilizado para los conjuntos de entrenamiento, validación y prueba, y que,

inclusive, fue grabado en una sesión diferente a la de los demás. Sobre dichos datos, se calculó la exactitud de las letras predichas (i.e., el número de letras correctamente predichas dividido el total de letras).

A continuación, se presentan diferentes cadenas de procesamiento implementadas, así como los resultados obtenidos para cada una de ellas. Todas estas cadenas fueron implementadas utilizando MATLAB y son agrupadas según el módulo de adquisición de señales de EEG utilizado. Además, las señales producidas por todas estas cadenas utilizaron el clasificador que fue entrenado siguiendo el procedimiento descrito en el capítulo anterior.

a. Cadenas implementadas sobre las señales adquiridas de bases de datos. En este caso, se utilizaron las grabaciones incluidas en la segunda edición de la competencia BCI, las cuales, como ya se presentó, fueron otorgadas por el centro Wadsworth. En el caso de estas señales, la cadena de procesamiento fue inspirada por el trabajo de Kaper [42], cuyo equipo fue ganador en la competencia BCI donde se utilizaron estos datos. En su trabajo, Kaper y su equipo [42] aplicaron la siguiente cadena de procesamiento: Se seleccionaron ciertos canales específicos, los cuales ocupaban posiciones occipitales, parietales y centrales. Sobre cada uno de estos canales, se aplicó un filtro pasa-banda de 0.5 a 30 Hz sobre las señales de EEG, las cuales luego fueron normalizadas para hallarse en un intervalo de [-1, 1]. Luego, se adquirían las muestras correspondientes a los 600 ms siguientes a una iluminación en pantalla. Los segmentos adquiridos de esta manera y correspondientes a la misma letra eran promediados y se alimentaba al clasificador con una muestra positiva por cada muestra negativa. Siguiendo la estructura anterior, se planteó la cadena que se presenta a continuación, la cual va seguida del Cuadro 3 que indica sus métricas de rendimiento. En dicho cuadro, la exactitud fue calculada sobre 31 letras.

1) Cadena de procesamiento A.1

- Se aplicó a las señales EEG un filtro pasa-banda de 0.5 a 30 Hz, de orden 120 y fase cero.
- Se aplicó una normalización temporal, por corrida, a unidades tipificadas z.
- Se toman todas las muestras, de todos los canales, de los 600 ms posteriores a la estimulación.
- Se entrenó y validó al clasificador.

Cuadro 3. Desempeño de las cadenas implementadas utilizando el señales adquiridas desde bases de datos

Cadena de procesamiento	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)
A.1	3	100	51.27	74.21	60.64	100

Como se puede observar en la descripción de la cadena de procesamiento A.1, en este caso no se estaba obteniendo la señal promedio para cada uno de los elementos de la matriz, sino que se entrenó y clasificó utilizando las señales sin promediar. Para predecir la letra, se obtenía cuál columna y cuál fila había tenido más clasificaciones positivas. Puesto que se contaban con 64 canales muestreados a 240 Hz y, originalmente, se estaban considerando todas las muestras provenientes de todos los canales, la dimensión de cada uno de

los vectores era muy grande, dificultando la tarea de clasificación y aumentando el tiempo requerido para entrenar y validar al algoritmo. Para reducir dicha dimensión, podría aplicarse una decimación y selección de canales. Otro factor que incrementaba la dificultad de la clasificación y el tiempo que esta tomaba era la gran cantidad de ejemplos que se tenían, debido a que estos no fueron promediados. Por otra parte, la exactitud del clasificador fue perfecta, lo cual transmitía la idea de que la cadena A.1 era válida. Sin embargo, existían varios métodos que lograban una clasificación perfecta utilizando este conjunto de datos [11]. Esto fue un indicador de que no era conveniente invertir muchos recursos para optimizar la cadena de procesamiento sobre estos datos, no solo por la facilidad que presentaban, sino también porque correspondía a sujetos y métodos de adquisición distintos a los que se utilizarían luego. Por esta razón, ya no se implementó una segunda cadena de procesamiento sobre estos datos.

b. Cadenas implementadas sobre las señales adquiridas con Emotiv EPOC y BCI2000. El siguiente paso fue adquirir grabaciones de EEG utilizando el Emotiv EPOC y BCI2000 sobre seis diferentes personas. De estas seis personas, solo una era mujer y las edades variaban entre los 23 y 53 años, siendo cinco de dichos sujetos de 23 años. Los parámetros utilizados en BCI2000 para el teclado de P300 fueron los mismos que los utilizados en el conjunto de datos de la competición BCI: Iluminaciones con una duración de 100 ms y un intervalo inter-estímulo de 75 ms, y 15 repeticiones de la iluminación de toda la matriz. Tales parámetros fueron utilizados de nuevo debido a los buenos resultados que se obtuvieron para la cadena de procesamiento A.1. La diferencia era entonces que ahora, por usarse el Emotiv, se contaba con 14 canales muestreados a 128 Hz. Además, se buscó realizar las grabaciones en lugares que no presentaran muchos distractores (e.g., ruido, lugares concurridos, etc.) para los sujetos de prueba.

Para cada uno de los sujetos, se grabaron cuatro sesiones. En cada una de dichas sesiones, se realizaron cinco corridas, conteniendo cada una de ellas tres letras. Esto quiere decir que, para cada usuario, se contaba con información relacionada a la estimulación de 60 letras. Los datos provenientes de las primeras tres sesiones fueron divididos, al azar, en tres conjuntos para entrenar, validar y obtener el puntaje F_1 del clasificador. Luego, dicho clasificador fue utilizado para predecir, fuera de línea, las letras de la cuarta sesión (15 letras) utilizando las 15 iluminaciones de la matriz disponibles para cada letra, de donde se obtuvo la exactitud del clasificador. Este procedimiento fue realizado tres veces para cada uno de los usuarios, debido a que la división al azar de los datos usados para el entrenamiento y validación provocaban diferentes exactitudes al predecir sobre la cuarta sesión. Una vez obtenidos los resultados para todos los sujetos, se utilizó como métrica final del desempeño del clasificador, la exactitud promedio para todos los usuarios.

A continuación, se presentan las diferentes cadenas de procesamiento implementadas, así como el rendimiento obtenido para cada una de ellas:

1) Cadena de procesamiento B.1

- Se aplicó a las señales EEG un filtro pasa-banda FIR de 0.5 a 30 Hz, de orden 120 y fase cero.
- Se aplicó una normalización temporal, por corrida, a unidades tipificadas z.
- Se toman todas las muestras, de todos los canales, de los 600 ms posteriores a la estimulación.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.
- Se entrenó y validó al clasificador.

Cuadro 4. Desempeño de la cadena de procesamiento B.1

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)
A.01	0.0001	10,000	73	78	76	66.67
	0.03	30	73	57	64	60.00
	0.01	300	80	86	83	93.33
A.02	1	300,000	93	93	93	80.00
	0.001	300	92	86	89	80.00
	0.0001	300	67	57	61	86.67
A.03	30	1,000	75	43	54	46.67
	30	1,000	53	57	55	26.67
	30	300	88	50	64	6.67
A.04	10	100	70	78	74	66.67
	3	100	75	67	70	73.33
	3	300,000	82	100	90	60.00
A.05	10	300	100	100	100	73.33
	0.0001	100	82	100	90	86.67
	0.0001	1,000	80	89	84	86.67
A.06	1	10,000	62	71	67	86.67
	0.03	30,000	86	43	57	60.00
	10	100	92	78	85	80.00
					Promedio	67.78

En la cadena de procesamiento B.1, se siguió la misma estructura de la cadena A1, con la diferencia de que se promediaron las señales correspondientes al mismo elemento de la matriz de estimulación. Esto se realizó para mejorar la relación de señal a ruido, así como para reducir el número de ejemplos que se utilizaban para clasificar. De esta forma, se reducía el tiempo requerido por la clasificación. En el Cuadro 4, se puede apreciar que la exactitud del clasificador varía considerablemente entre sujetos, mientras que también existen variaciones notorias dentro de los resultados obtenidos para un mismo sujeto. Además, cabe mencionar que, puesto que las exactitudes fueron calculadas con respecto a únicamente 15 letras, cada letra fallida significa una disminución del 6.67% en la exactitud, por lo que consideraremos una exactitud por encima del 80% como una adecuada, ya que en dicho caso, únicamente se fallaron 3 de 15 letras. Sin embargo, este definitivamente no es el caso para la cadena de procesamiento B.1 lo cual indica la necesidad de mejorar la relación de señal a ruido de los datos entregados al clasificador.

2) Cadena de procesamiento B.2

- Se aplicó un filtro espacial CAR a las señales EEG.
- Se aplicó a las señales EEG un filtro pasa-banda FIR de 0.5 a 30 Hz, de orden 120 y fase cero.
- Se aplicó una normalización temporal, por corrida, a unidades tipificadas z.
- Se toman todas las muestras, de todos los canales, de los 800 ms posteriores a la estimulación.
- Se aplicó una decimación con un factor de dos.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.
- Se entrenó y validó al clasificador.

Cuadro 5. Desempeño de la cadena de procesamiento B.2

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)
A.01	0.0001	1,000	67	86	75	80.00
	0.03	300	80	86	83	93.33
	0.001	30	50	43	46	93.33
A.02	0.0001	3,000	69	78	73	80.00
	0.0001	1,000	76	93	84	80.00
	0.001	100	92	78	85	86.67
A.03	10	100	89	57	70	53.33
	0.3	1,000	88	50	64	60.00
	0.03	300	75	64	69	66.67
A.04	0.0001	300,000	80	44	57	73.33
	3	100,000	100	78	88	73.33
	0.0001	1,000	62	56	59	73.33
A.05	0.0001	300	100	100	100	93.33
	0.0001	100	82	100	90	93.33
	0.0001	300	100	100	100	80.00
A.06	10	100	78	78	78	73.33
	0.03	3,000	91	71	80	73.33
	0.03	3,000	73	57	64	86.67
					Promedio	78.52

En la cadena B.2, se han introducido varias diferencias con respecto a la B.1: Se añadió un filtro espacial CAR previo a cualquier otra operación; luego, se consideraron 800 ms posteriores al estímulo en lugar de 600 ms y se agregó una decimación con un factor de dos. La aplicación del filtro CAR se realizó para mejorar la relación de señal a ruido de los datos recuperados, mientras que la decimación con un factor de dos fue para reducir la dimensión de los ejemplos por la mitad, para facilitar la tarea de clasificación. Nótese que, puesto que las señales de EEG ya se hallaban banda-limitadas a 30 Hz y la decimación aplicada correspondía a un re-muestreo de la señal con una frecuencia de 64 Hz (para lo cual la señal a decimar debería de estar banda-limitada a 32 Hz, a lo sumo), el procedimiento para aplicar la decimación consistía únicamente en tomar una de cada dos muestras; i.e., la etapa de filtrado requerida para la decimación ya había sido aplicada anteriormente con el objetivo de retirar las componentes frecuenciales que representaban ruido para esta aplicación. Con estas modificaciones se logró elevar el promedio de la exactitud en un 10.74%, como se observa en el Cuadro 5. Otro aspecto a resaltar es que las exactitudes para un mismo sujeto presentan una

menor variación que en el Cuadro 4, lo cual indica que con la cadena B.2 se ha reducido el efecto del ruido más que con la cadena B.1, ya que la clasificación es menos sensible a la separación de los datos realizada para entrenar y validar al clasificador: i.e., las señales ahora presentan características que facilitan su clasificación, mientras que la presencia de aquellas que dificultaban dicha tarea se ha visto atenuada.

3) Cadena de procesamiento B.3

- Se aplicó un filtro espacial CAR a las señales EEG.
- Se aplicó a las señales EEG un filtro pasa-banda FIR de 1 a 15 Hz, de orden 120 y fase cero.
- Se toman todas las muestras, de todos los canales, de los 800 ms posteriores a la estimulación.
- Se aplicó una decimación con un factor de dos.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.
- Se aplicó una normalización de características por corrida, a unidades tipificadas z.
- Se entrenó y validó al clasificador.

Cuadro 6. Desempeño de la cadena de procesamiento B.3

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)
A.01	0.0003	1,000	68	78	73	73.33
	0.003	30,000	75	86	80	80.00
	1	1,000	92	86	89	86.67
A.02	3	3,000	100	71	83	80.00
	0.003	1,000	80	86	83	86.67
	0.001	300	92	78	85	86.67
A.03	30	10,000	100	78	88	86.67
	3	3,000	85	79	81	93.33
	10	3,000	82	100	90	80.00
A.04	0.01	100,000	75	100	86	80.00
	0.003	30,000	88	78	82	86.67
	10	1,000	89	89	89	73.33
A.05	0.0003	1,000	100	100	100	93.33
	0.0001	3,000	90	100	95	93.33
	0.0001	1,000	100	100	100	93.33
A.06	0.001	300	80	86	83	66.67
	0.0001	3,000	90	64	75	80.00
	0.0001	1,000	89	57	70	73.33
					Promedio	82.96

En el caso de la cadena B.3, se introdujeron dos modificaciones con respecto a la anterior: Primero se filtró de 1 hasta 15 Hz y luego, la normalización no fue realizada temporalmente, sino por características, el cual es uno de los métodos recomendados para operar con algoritmos de aprendizaje automático [61]. Para realizar dicha normalización, se tomaron en cuenta todos los ejemplos pertenecientes a una misma corrida. Se puede apreciar en el Cuadro 6 que, con estos cambios, se logró un incremento del 4.44% en la exactitud promedio, con respecto al caso anterior.

4) Cadena de procesamiento B.4

- Se aplicó un filtro espacial CAR a las señales EEG.
- Se aplicó a las señales EEG un filtro Butterworth pasa-banda IIR de 1 a 30 Hz, de orden 6.
- Se toman todas las muestras, de todos los canales, de los 800 ms posteriores a la estimulación.
- Se aplicó una decimación con un factor de dos.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.
- Se aplicó una normalización de características por corrida, a unidades tipificadas z.
- Se entrenó y validó al clasificador.

Cuadro 7. Desempeño de la cadena de procesamiento B.4

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)
A.01	0.01	3,000	57	86	86	80.00
	0.0003	300	47	50	48	66.67
	0.001	300	67	71	69	73.33
A.02	0.001	300	87	93	90	93.33
	3	1,000	75	86	80	80.00
	0.001	1,000	100	100	100	86.67
A.03	10	30,000	85	78	81	73.33
	0.0001	10,000	76	93	84	73.33
	0.01	30,000	77	71	74	66.67
A.04	0.0001	300	45	56	50	53.33
	0.001	300	100	44	62	60.00
	10	3,000	75	67	70	66.67
A.05	0.0001	3,000	75	100	86	86.67
	0.0001	300	89	89	89	73.33
	0.001	100	82	100	90	86.67
A.06	0.001	1,000	85	78	81	80.00
	3	1,000	100	86	92	66.67
	0.003	300	92	86	89	73.33
					Promedio	74.44

En las cadenas implementadas hasta ahora, se utilizaron filtros FIR. Dichos filtros serían difíciles de implementar en una aplicación de tiempo real, especialmente por el elevado orden de estos, lo cual indica que requieren un gran número de operaciones y recursos de memoria. Por esta razón, en la cadena de procesamiento B.4 se introdujo el uso de un filtro IIR que filtraba de 1 a 30 Hz. Como se observa en el Cuadro 7, la exactitud promedio disminuyó 8.52% con respecto a la cadena B.3. Por una parte, esto ocurrió por ser las características del filtro IIR no tan cercanas a las ideales como las del filtro FIR de orden elevado y fase cero usado previamente. Por otra parte, en esta cadena se regresó a utilizar una banda de paso de 1 a 30 Hz, cuando en la cadena B.3 se había utilizado una de 1 a 15 Hz. Esto sugeriría que la segunda banda de paso proporciona mejores resultados.

5) Cadena de procesamiento B.5

- Se aplicó un filtro espacial CAR a las señales EEG.
- Se aplicó a las señales EEG un filtro Butterworth pasa-banda IIR de 1 a 30 Hz, de orden 6.
- Se aplicó una normalización espacial, por corrida, a unidades tipificadas z.
- Se toman todas las muestras, de todos los canales, de los 800 ms posteriores a la estimulación.
- Se aplicó una decimación con un factor de dos.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.
- Se entrenó y validó al clasificador.

Cuadro 8. Desempeño de la cadena de procesamiento B.5

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)
A.01	0.0001	100	80	86	83	86.67
	0.1	30	100	86	92	86.67
	0.003	30	72	93	81	93.33
A.02	0.003	10	78	78	78	86.67
	10	300	86	86	86	93.33
	0.001	30	86	86	86	86.67
A.03	3	100	85	79	81	93.33
	0.001	30	75	64	69	80.00
	10	300	92	86	89	86.67
A.04	0.0003	100	80	89	84	80.00
	0.003	30	100	78	88	80.00
	30	1,000	69	100	82	86.67
A.05	0.0001	100	100	100	100	86.67
	0.0003	100	100	100	100	80.00
	0.0001	30	100	89	94	80.00
A.06	0.0001	1,000	81	93	87	80.00
	0.003	10	100	57	73	66.67
	30	1,000	93	93	93	60.00
					Promedio	82.96

En la cadena de procesamiento B.5, se deseaba explorar el efecto que podría tener el método de normalización sobre los resultados, razón por la cual el único cambio realizado con respecto a la cadena B.4 es la normalización, que fue realizada espacial y por corrida. Se puede observar en el Cuadro 8 que con esta cadena se obtuvo un incremento del 8.52% en la exactitud promedio, igualando el desempeño mostrado por la cadena B.3. Además de esta mejora en el desempeño, la normalización espacial presenta otra ventaja frente a la normalización de características por corrida en cuanto a la implementación en tiempo real: La normalización de características por corrida requeriría de datos correspondientes a, al menos, tres letras para poder ser calculadas, lo cual implica un mayor consumo de memoria almacenando las características previas. Además, al querer normalizar las primeras dos letras, no se podrá aplicar este método, puesto que no se dispone aún de la información necesaria. Esto afectaría negativamente a las primeras dos predicciones, por lo que se desea tener un método que no dependa de datos externos a los de la estimulación de la letra actual para evitar esto. Una solución a dicho problema es la propuesta por la cadena de procesamiento B.5, en la

cual la normalización se hace de forma espacial: De esta manera, para normalizar cada dato, no se deben de recopilar un conjunto de estos a través del tiempo, sino que la normalización puede realizarse cada vez que se muestrea, puesto que en cada muestreo, se obtienen datos correspondientes a los 14 canales que permiten el cálculo deseado.

6) Cadena de procesamiento B.6

- Se aplicó un filtro espacial CAR a las señales EEG.
- Se aplicó a las señales EEG un filtro pasa-banda IIR de 1 a 15 Hz, de orden 6.
- Se aplicó una normalización espacial, por corrida, a unidades tipificadas z.
- Se toman todas las muestras, de todos los canales, de los 800 ms posteriores a la estimulación.
- Se aplicó una decimación con un factor de dos.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.
- Se entrenaron y validaron tres clasificadores, los cuales fueron utilizados para realizar tres clasificaciones individuales, así como también para realizar una sola clasificación usando las salidas de los tres.

Cuadro 9. Desempeño de la cadena de procesamiento B.6

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)	Exactitud con tres clasificadores (%)
A.01	0.001	10	91	71	80	86.67	80.00
	0.1	30,000	87	93	90	80.00	
	30	300	70	100	82	86.67	
A.02	0.1	3,000	74	100	85	86.67	86.67
	0.0001	100	80	86	83	93.33	
	3	100	91	71	80	80.00	
A.03	0.001	300	78	78	78	53.33	80.00
	0.001	100	73	78	76	80.00	
	0.003	100	91	71	80	73.33	
A.04	10	1,000	88	78	82	66.67	66.67
	0.003	100	82	100	90	66.67	
	0.001	30	75	67	70	66.67	
A.05	0.0001	100	80	89	84	80.00	80.00
	10	100	100	100	100	86.67	
	0.001	30	100	89	94	80.00	
A.06	0.003	10	92	78	85	73.33	86.67
	0.003	30	100	100	100	86.67	
	3	100	87	93	90	86.67	
					Promedio	78.52	80.00

En la cadena B.6 se cambió la banda de paso del filtro de IIR de tal forma que la frecuencia de corte se encontraba en 15 Hz y no en 30 Hz, como se había realizado en la cadena B.5. Contrario a lo anteriormente sospechado de que utilizar una frecuencia de corte de 15 Hz era mejor que una de 30 Hz, la exactitud promedio disminuyó en un 4.44%, como se puede observar en el Cuadro 9.

Además, en esta cadena de procesamiento se agregó un paso más: Al estar verificando la exactitud de cada una de las cadenas implementadas anteriormente, se observó que, para un mismo sujeto pero utilizando diferentes clasificadores, a veces algunos clasificadores acertaban en letras en los cuales los otros fallaban. Esto generó la duda de si se podía, de alguna manera, combinar las salidas de los tres clasificadores para tener mejores resultados. Al investigar, se halló que Rakotomamonjy y Guide [72] propusieron realizar un ensamble de clasificadores SVM, cada uno de los cuales eran entrenados sobre un subconjunto diferente de los datos de entrenamiento (siendo dichos subconjuntos disjuntos entre sí) y, posteriormente, se utilizaban todas sus salidas para realizar una predicción. En este caso, como ya se contaban con diferentes clasificadores entrenados sobre diferentes selecciones del conjunto de entrenamiento (los cuales, a diferencia del método anterior no eran disjuntos entre sí), se procedió a sumar las probabilidades obtenidas a la salida del clasificador. Luego, se seleccionaba la columna y fila que presentaban la mayor probabilidad de corresponder a la reacción a un estímulo positivo, y con dicha información se presentaba la letra predicha. Se puede observar en el Cuadro 9 que esta técnica permitió un incremento en la exactitud promedio de los clasificadores.

7) Cadena de procesamiento B.7

- Se aplicó un filtro espacial CAR a las señales EEG.
- Se aplicó a las señales EEG un filtro pasa-banda IIR de 1 a 15 Hz, de orden 4.
- Se aplicó una normalización espacial, por corrida, a unidades tipificadas z.
- Se toman todas las muestras, de todos los canales, de los 800 ms posteriores a la estimulación.
- Se aplicó una decimación con un factor de dos.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.
- Se entrenaron y validaron tres clasificadores, los cuales fueron utilizados para realizar tres clasificaciones individuales, así como también para realizar una sola clasificación usando las salidas de los tres.

La única diferencia de esta cadena de procesamiento B.7 con respecto a la anterior es que el filtro IIR aplicado fue de orden 4 y no de orden 6. Este cambió logró un incremento del 2.59% en la exactitud promedio de los clasificadores individuales y del 5.56% en el ensamble de tres clasificadores, como se puede observar en el Cuadro 10. Además, se puede observar que la exactitud promedio de las clasificaciones individuales de esta cadena es mayor al de todas las cadenas implementadas anteriormente, siendo las únicas excepciones las cadenas B.3 y B.5. Sin embargo, al utilizar el ensamble de clasificadores, se obtiene una mejor exactitud promedio.

Cuadro 10. Desempeño de la cadena de procesamiento B.7

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)	Exactitud con tres clasificadores (%)
A.01	0.003	10	81	93	87	80.00	86.67
	0.001	10	67	86	75	73.33	
	0.003	100	80	86	83	93.33	
A.02	0.001	100	100	93	96	86.67	86.67
	0.001	300	75	86	80	86.67	
	0.003	100	76	93	84	93.33	
A.03	30	1,000	83	71	77	66.67	80.00
	0.001	100	100	78	88	73.33	
	0.003	1,000	78	78	78	66.67	
A.04	0.003	100	73	89	80	73.33	80.00
	0.003	10	78	78	78	60.00	
	0.003	100	82	100	90	73.33	
A.05	1	100	100	88	94	93.33	86.67
	0.0001	100	90	100	95	93.33	
	0.001	100	89	89	89	86.67	
A.06	0.001	100	80	86	83	93.33	93.33
	0.0001	100	92	86	89	80.00	
	3	100	90	64	75	86.67	
					Promedio	81.11	85.56

8) Cadena de procesamiento B.8

- Se aplicó un filtro espacial CAR a las señales EEG.
- Se aplicó a las señales EEG un filtro pasa-banda IIR de 1 a 15 Hz, de orden 4.
- Se aplicó una normalización temporal, por corrida, a unidades tipificadas z.
- Se toman todas las muestras, de todos los canales, de los 800 ms posteriores a la estimulación.
- Se aplicó una decimación con un factor de dos.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.
- Se entrenaron y validaron tres clasificadores, los cuales fueron utilizados para realizar tres clasificaciones individuales, así como también para realizar una sola clasificación usando las salidas de los tres de ellos.

En la cadena de procesamiento B.8, se cambió la normalización espacial por corrida a una temporal por corrida. Se hizo esto puesto que en la literatura se presenta la normalización temporal y por características, pero nunca se explora una normalización espacial. Por consiguiente, se deseaba verificar que ésta última normalización sí presentaba ventaja sobre la temporal. (La ventaja sobre la normalización por características ya se había observado al comparar la cadena B.4 con la B.5). Al observar el Cuadro 11, se comprueba que la normalización espacial presenta ventajas sobre la temporal, puesto que ambas exactitudes fueron mayores en el caso de la cadena B.7 que en la B.8.

Cuadro 11. Desempeño de la cadena de procesamiento B.8

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)	Exactitud con tres clasificadores (%)
A.01	0.0001	300	78	78	78	80.00	86.67
	10	100	77	71	74	86.67	
	0.003	3,000	60	93	72	86.67	
A.02	0.03	3,000	65	78	71	86.67	86.67
	0.001	10	77	71	74	86.67	
	0.0001	30,000	72	93	81	86.67	
A.03	30	1,000	90	64	75	80.00	86.67
	0.01	1,000	82	64	72	80.00	
	0.03	3,000	73	78	76	80.00	
A.04	10	300	75	33	46	66.67	60.00
	0.1	30,000	82	100	90	60.00	
	0.001	100	100	67	80	73.33	
A.05	0.0001	100	100	100	100	93.33	93.33
	0.0003	100	90	100	95	86.67	
	0.0001	3,000	90	100	95	93.33	
A.06	3	30	76	93	84	66.67	80.00
	30	1,000	82	64	72	73.33	
	0.001	100	80	86	83	73.33	
					Promedio	80.00	82.22

c. Cadenas implementadas sobre las señales adquiridas con Emotiv EPOC y el módulo de obtención de señales. Debido a que esto corresponde a la integración entre los dos módulos de la interfaz cerebro computadora, será presentada posteriormente, luego de analizar al segundo módulo.

B. MÓDULO DE OBTENCIÓN DE SEÑALES EEG Y DESARROLLO DE APLICACIÓN PARA INTERFAZ CEREBRO COMPUTADORA

1. Obtención de señales EEG

a. Diseño experimental. Se utilizaron dos equipos para obtención de señales electroencefalográficas, el primero fue el gorro de electrodos de *Electro-Cap International* y el segundo fue el Emotiv EPOC de Emotiv *Systems*. Se deseaba determinar cuál de los dos equipos proporcionaba una mejor calidad de señales EEG, por lo que se realizaron pruebas con ambos y se compararon los resultados obtenidos.

Se investigó el tipo de ruido e interferencias a las que están expuestas las señales EEG y así, determinar qué filtros serían necesarios para limpiarlas. También se investigó sobre qué filtros se han utilizado antes y cuáles son los más recomendados.

Se diseñaron los filtros analógicos en *Multisim* y se determinó si la respuesta de los filtros era la deseada o no. Se conectó el gorro a los filtros y se utilizó el *myDAQ* de *National Instruments* para poder observar las señales en la computadora utilizando MATLAB. Se expuso al usuario a estímulos auditivos y visuales y se trató de encontrar algún patrón en las señales EEG captadas en la computadora.

Posteriormente se deseaba probar el Emotiv EPOC. Este equipo posee un filtrado interno por lo que no era necesario aplicar filtros analógicos. Se utilizó el programa que se proporciona al comprar la licencia de Emotiv. Dicho programa permite mover objetos en la computadora utilizando las señales EEG. Se entrenó al sistema y se probó su eficiencia. Se compararon las respuestas de los dos equipos y se determinó cuál proporcionaba mejores resultados.

También se investigó sobre herramientas de MATLAB que permitieran accesar a los canales del Emotiv EPOC. De esta forma se podría programar la captación de señales EEG directamente en MATLAB.

b. Resultados

1) Diseño de filtros analógicos. Como se mencionó anteriormente, para la obtención de señales electroencefalográficas se utilizó el gorro de electrodos de *Electro-Cap International* y el gel conductor ECI Electro-Gel (Figura 70). Posteriormente se utilizó el Emotiv EPOC para determinar cuál de los dos equipos proporcionaba señales EEG de mejor calidad.

Figura 70**.** *Electro-Cap* y gel conductor.

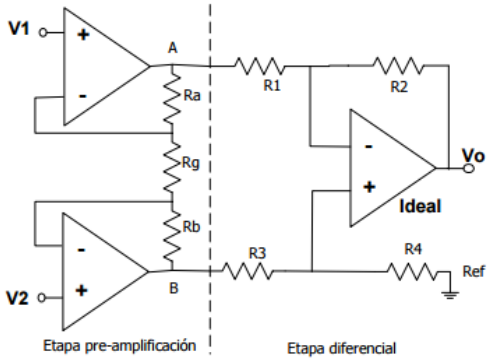
Debido a que la magnitud de las señales eléctricas cerebrales se encuentran en un rango menor a 100μV, es necesario acondicionar la señal para que sea compatible con diferentes dispositivos, tales como: computadoras y microcontroladores, entre otros. Los amplificadores adecuados para EEG deben amplificar solamente la señal fisiológica y atenuar ruido y señales de interferencia [89].

La señal de entrada a los amplificadores consiste de cinco componentes: señales biopotenciales deseadas y no deseadas, ruido debido a la fuente de alimentación (50/60 Hz y sus armónicos), ruido generado por el contacto entre la piel y los electrodos (60 Hz), por ultimo cualquier otro ruido que puede generarse en el

circuito. Para proporcionar una señal de buena calidad y que posee un nivel de voltaje adecuado para ser procesado, se requiere una amplificación entre 100 y 100,000. Se debe procurar mantener la mejor relación entre la señal deseada y el ruido en el sistema [89]. En otras palabras, no necesariamente amplificar 100,000 veces la señal de entrada produce la mejor señal de salida porque aparte de amplificar las señales biopotenciales de interés, se amplifica el ruido también. Por lo tanto es necesario utilizar una amplificación intermedia que minimice el ruido y maximice la señal de interés.

Como primera etapa para la obtención de señales EEG, se utilizó un amplificador instrumental. La amplificación se realiza comparando el voltaje entre dos electrodos diferentes. Además, las características de este tipo de amplificador es que se obtiene la mejor razón entre señal de interés y ruido, es decir, se minimiza el ruido y se maximiza la amplificación de las señales cerebrales [55].

Figura 71. Amplificador instrumental [55].



La etapa de pre-amplificación aumenta la impedancia de entrada del conjunto y amplifica las señales entrantes. La etapa diferencial realiza la resta entre las dos señales y la salida Vo muestra el resultado de esta operación. Es necesario que todas las resistencias de la etapa diferencial sean iguales para asegurar que si V1 y V2 son iguales, la salida debe ser 0V. Por lo tanto, la función que describe la salida, Vo, en términos de las dos entradas V1 y V2 es:

$$V_O = \left(1 + \frac{R_a + R_b}{R_g}\right)(V_1 - V_2) + V_{ref}$$

Igualando Ra y Rb se obtiene:

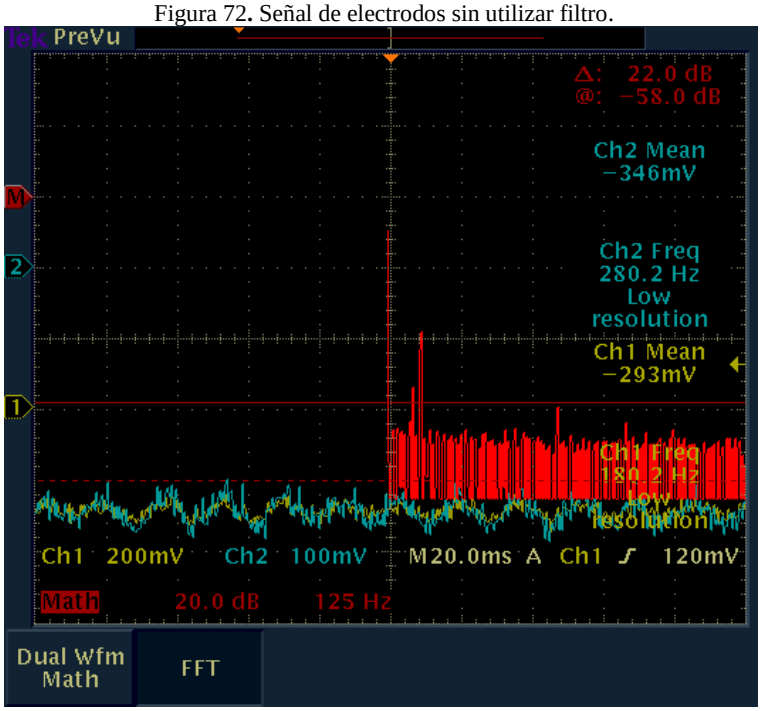
$$V_O = \left(1 + 2\frac{R_a}{R_g}\right)(V_1 - V_2) + V_{ref}$$

Por lo tanto se puede modificar la amplificación modificando el valor de R_g. Entre más pequeño sea este valor, mayor será la amplificación observada. Sin embargo, no es aconsejable amplificar demasiado la señal porque se puede saturar el amplificador operacional. Para el circuito se utilizó $R_a = R_b = R_1 = R_2 = R_3 = R_4 = 10k\Omega$ para poder variar Rg entre 10Ω y $1k\Omega$ para obtener una ganancia de 1000 a 10, respectivamente utilizando un potenciómetro de precisión de $1k\Omega$ para R_g.

Como se mencionó anteriormente las frecuencias de las señales cerebrales de interés se encuentran entre 0.2 a 30 Hz. Por ende es necesario utilizar filtros pasa bajos para eliminar todas las frecuencias superiores a 30 Hz, especialmente el ruido debido a la alimentación eléctrica, el cual posee una frecuencia de 60 Hz.

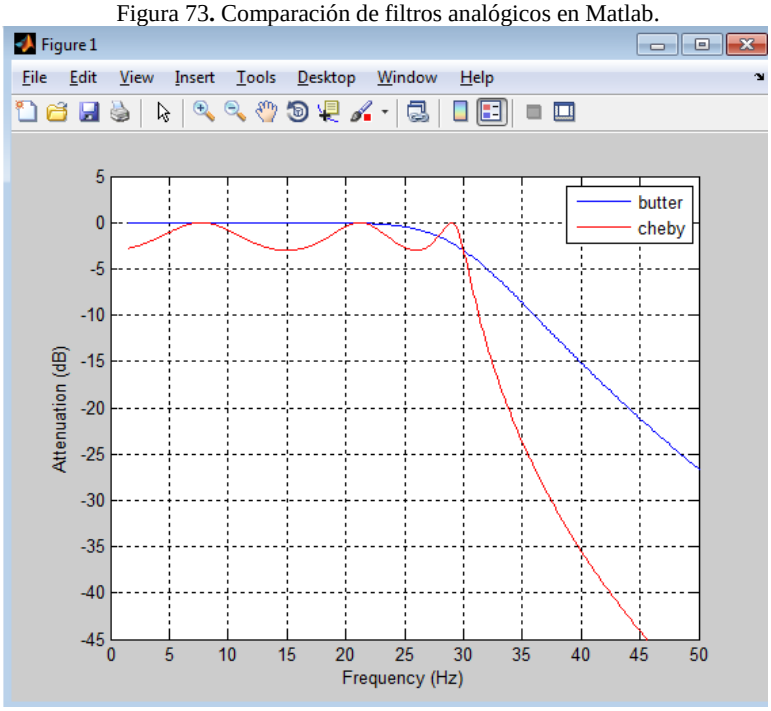
En la Figura 72 se muestra la señal obtenida con el gorro de electrodos sin utilizar algún filtro. Se aplicó la transformada rápida de Fourier para determinar qué frecuencias componen dicha señal. Como se puede observar, la frecuencia que mayor magnitud es la de 60 Hz. Esto demuestra que existe ruido de 60 Hz debido al contacto entre el cuero cabelludo y el electrodo.

Figura 72. Señal de electrodos sin utilizar filtro.



Para determinar qué filtro utilizar se simuló en MATLAB la respuesta de un filtro Butterworth y de un filtro Chebyshev, ambos con frecuencia de corte de 30 Hz y de 6 polos.

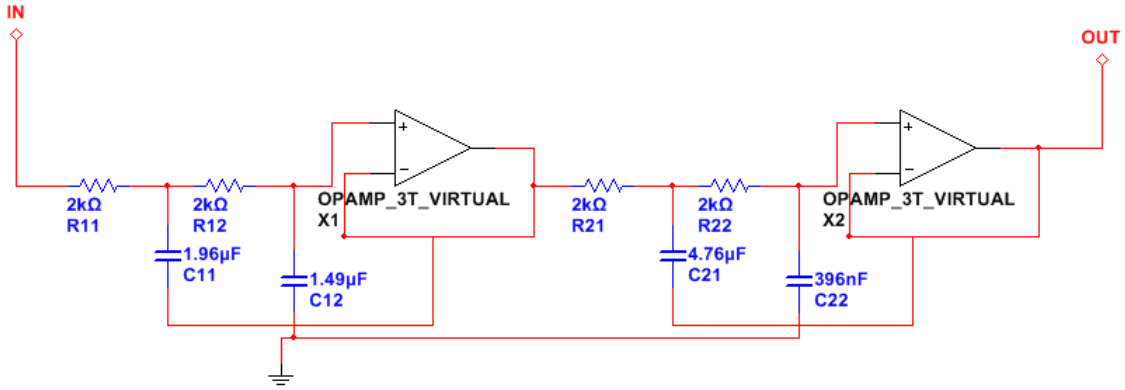
Figura 73. Comparación de filtros analógicos en Matlab.



El filtro Butterworth posee una ganancia constante antes de los 30 Hz pero su región de transición es mayor al del filtro Chebyshev. Sin embargo, la ganancia en la banda de paso del filtro Chebyshev oscila bastante.

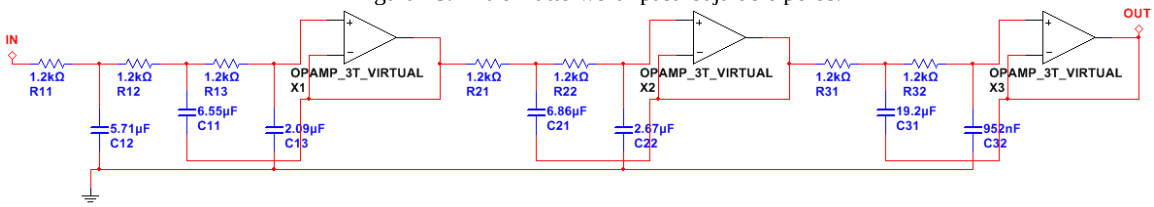
Para eliminar el ruido en la señal EEG, se diseñó un filtro pasa-baja de 60 Hz. Se escogió un filtro pasa-baja tipo Butterworth ya que este tipo de filtro mantiene una ganancia constante en toda la banda de paso y no alteraría las señales EEG captados con el gorro de electrodos [5]. Todos los filtros diseñados en este proyecto se realizaron utilizando la herramienta de *MultiSim*.

Figura 74. Filtro Butterworth pasa-baja de 4 polos.



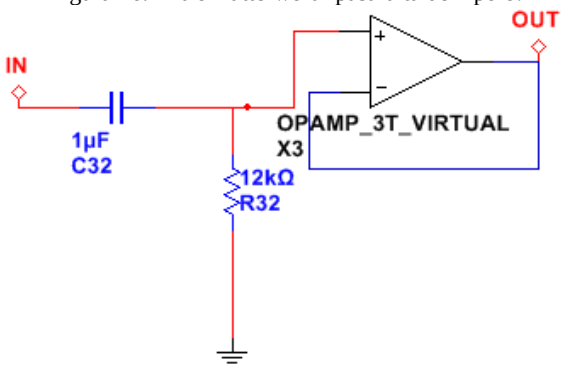
Posteriormente se construyó un filtro pasa-baja de 31 Hz para conectarlo en la salida del filtro anterior y eliminar todas las frecuencias que no se utilizan en la aplicación de P300.

Figura 75. Filtro Butterworth pasa-baja de 6 polos.



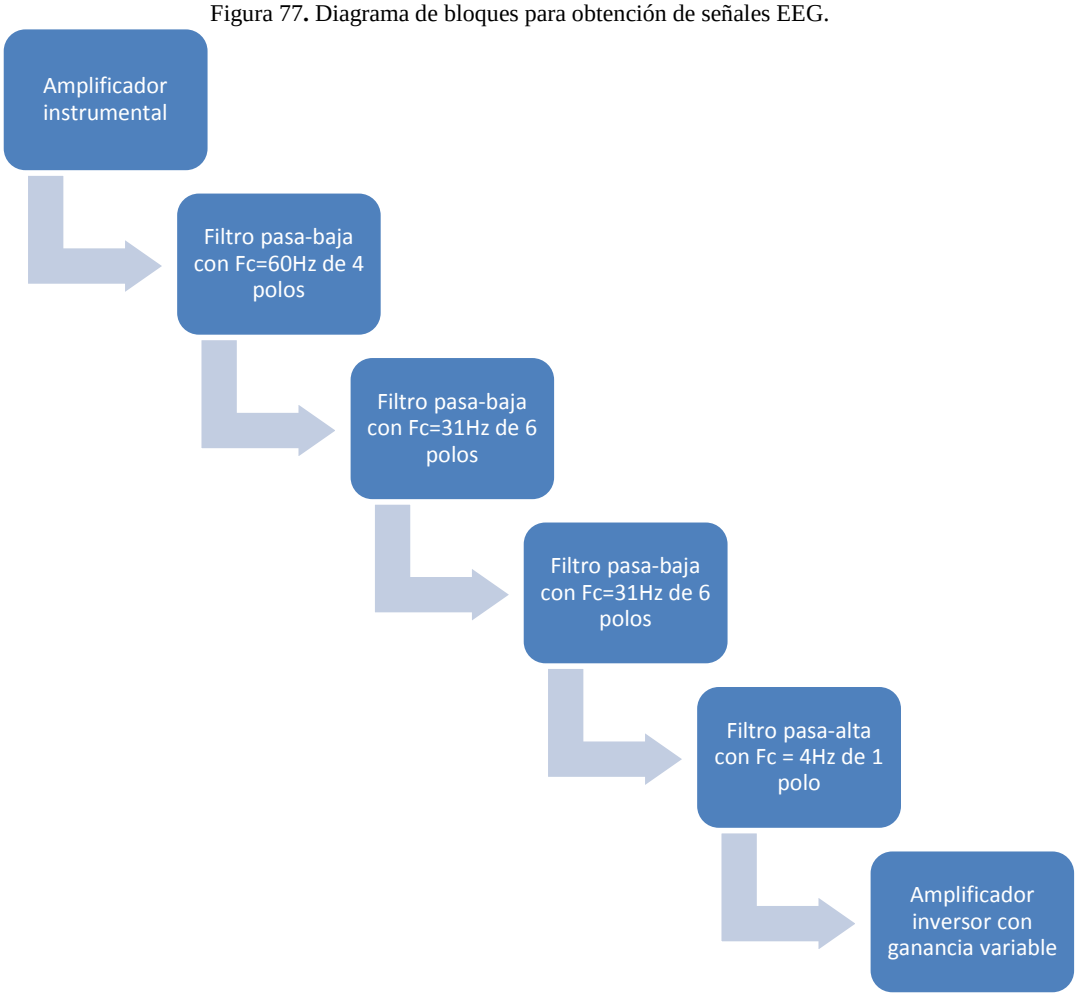
Se conectó dos de los filtros pasa-bajos de 31 Hz en cascada, formando un filtro pasa-baja de 12 polos. De esta forma, se podía asegurar la atenuación de todas las señales arriba de 31 Hz. Para eliminar los componentes con frecuencias debajo de 0.2 Hz, se construyó un circuito pasa-alta Butterworth de 4 Hz. Se utilizó un filtro con un polo porque no es crítico que se eliminen las frecuencias debajo de 0.2Hz, solamente se desea quitar cualquier voltaje DC que puedan poseer las señales EEG obtenidas con el gorro de electrodos.

Figura 76. Filtro Butterworth pasa-alta de 1 polo.



2) Implementación y pruebas de filtros diseñados. Para medir las señales en cada electrodo se utilizó la siguiente conexión de circuitos:

Figura 77. Diagrama de bloques para obtención de señales EEG.



Se empezó haciendo pruebas con dos de los 19 electrodos del gorro de electrodos de *Electro-Cap International*. Posteriormente se replicó el mismo circuito para incluir por lo menos dos pares de electrodos y así determinar si se puede detectar algún tipo de patrón en las señales o no.

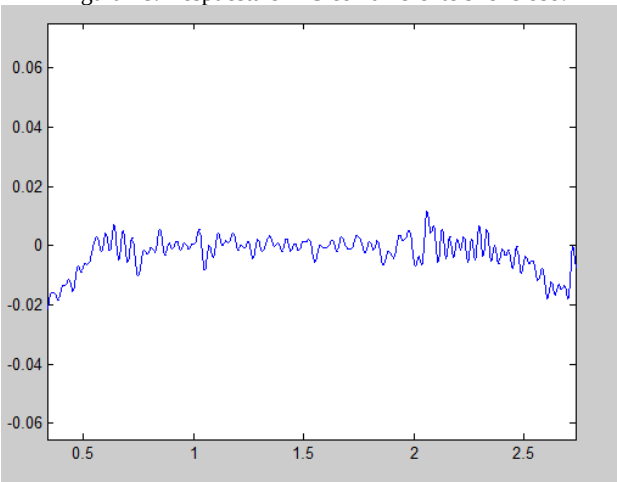
Para probar si había alguna reacción en las señales EEG ante un estímulo externo, se desarrolló un programa en MATLAB para grabar las señales de dos diferentes entradas analógicas. Para lograr captar las señales EEG en la computadora se utilizó un *myDAQ*, de *National Instruments*, el cual cuenta con dos entradas analógicas. Este equipo fue adquirido dentro de la universidad, en el departamento de ingeniería electrónica. Matlab posee librerías que permiten comunicarse con el *myDAQ* y accesar todas sus entradas analógicas y digitales [50].

Primero se tuvo que conectar el *myDAQ* a la computadora y entrar a MATLAB como administrador para registrarlo como componente de entrada y así poder utilizar las diferentes funciones de su librería. Una vez registrada solamente se configuraba la frecuencia de muestreo, el canal de entrada, el rango de voltaje de entrada y la cantidad de muestras a tomar.

Los electrodos que se utilizaron en esta etapa fueron T3 y O1, las cuales corresponden al lóbulo temporal izquierdo y occipital izquierdo. El lóbulo occipital está relacionado con aspectos visuales, por lo que se trató de exponer al usuario a diferentes estímulos visuales, como luz o ver algún objeto en específico. También se probó cerrar los ojos ya que las señales EEG presentan mayor amplitud cuando los ojos se encuentran cerrados [19].

Posteriormente se trató de tomar señales en el canal T3. Debido a que en el lóbulo temporal interviene el estímulo auditivo, se tomaron mediciones de este electrodo ante estímulos auditivos distintos. La primera prueba realizada consistía en estar en completo silencio mientras se realizaba la grabación. Los resultados se muestran en la Figura 78:

Figura 78**.** Respuesta en T3 con ambiente silencioso.



La segunda prueba para el lóbulo temporal consistía en tronar los dedos cerca de la oreja izquierda y tratar de detectar algún patrón en la señal grabada. Esta prueba se realizó varias veces y se intentó modificar el tiempo entre cada estímulo auditivo, de esta forma se podría comprobar que los patrones detectados no se debían a ruido en el ambiente.

Figura 79**.** Respuesta en T3 con 3 estímulos auditivos.

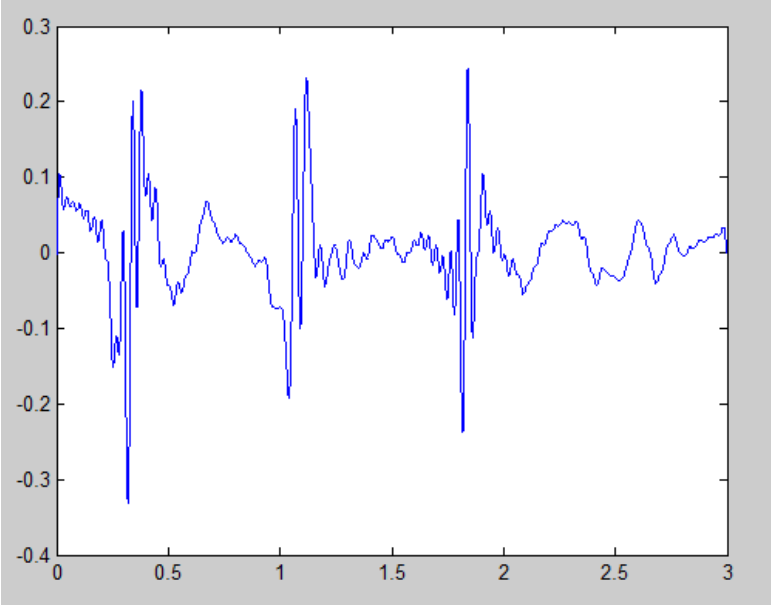
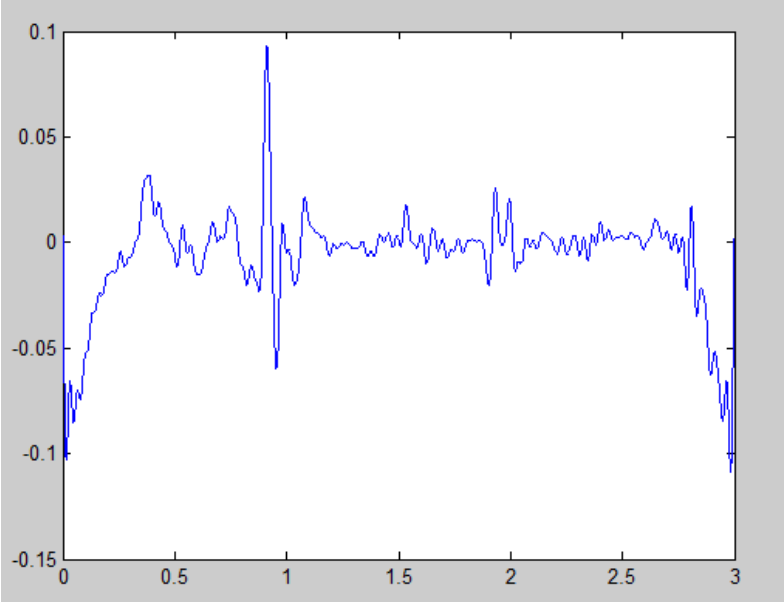


Figura 80**.** Respuesta en T3 con 2 estímulos auditivos.



3) Pruebas con Emotiv EPOC. Para las pruebas con el Emotiv EPOC se utilizó el programa suministrado por Emotiv *Systems* con la licencia del EPOC. Dicho programa se llama *Emo Engine* y cuenta con diferentes modos de operación: la configuración del casco, modo expresivo, modo afectivo, modo cognitivo y emulador de mouse [22].

La pestaña de configuración sirve para poder ver dónde se deben ubicar los electrodos, poder colocar el casco adecuadamente y determinar la calidad de las señales en cada electrodo. El modo expresivo consiste en una interfaz gráfica que demuestra las expresiones que el usuario realiza en tiempo real. El modo afectivo detecta estados de ánimo en la persona; puede determinar si la persona está calmada, pensativa, desinteresada o emocionada. El emulador de mouse utiliza el giroscopio del EPOC para mover el mouse con los movimientos de la cabeza [22].

El modo cognitivo es el que se utilizó y consiste en mover un cubo con pensamientos. Es necesario entrenar el sistema e indicar la acción a realizarse con cada pensamiento. Para todos los entrenamientos, el usuario tiene libertad con respecto a qué pensar, solamente debe mantenerse concentrado durante 8 segundos. Para volver a efectuar la acción, simplemente debe volver a pensar en lo que utilizó para entrenar cada acción [22].

Figura 81. Modo cognitivo en panel de control de EPOC [22].

Como paso inicial, se debe entrenar el estado neutral y posteriormente se debe entrenar cada acción que se desea realizar. Uno puede elegir qué acción desea que se realice. Entre las acciones que se pueden ejecutar está: rotar el cubo, empujarlo hacia el frente o hacia atrás, trasladarlo hacia la derecha, izquierda, arriba o abajo [22].

Se selecciona la acción a realizar y el usuario debe pensar en algo concreto, por ejemplo, un número, un color, una palabra, etc. Una vez entrenado, el cubo deberá realizar la acción que se configuró cada vez que el usuario vuelva a pensar lo que mismo que en entrenamiento. Se puede crear diferentes usuarios para guardar las configuraciones para cada persona de forma independiente.

Se realizaron pruebas del Emotiv EPOC con 6 personas diferentes para comprobar su funcionamiento. Todos los participantes lograron entrenar el sistema y mover el cubo. Se observó que el sistema carecía de robustez porque a veces surgían movimientos involuntarios del cubo o se dificultaba realizar las acciones deseadas.

4) Obtención de señales EEG con Emotiv EPOC. Emotiv *Systems* desarrolló un programa para obtener señales de todos los canales del EPOC desde Java [24]. Sin embargo, el programa de deletreo se empezó a programar en MATLAB por lo que se trató de encontrar formas de poder enviar y recibir información de MATLAB a Java. También se investigó sobre herramientas en MATLAB que permitieran accesar a los canales del EPOC directamente y así desarrollar todo el programa en MATLAB, tanto la interfaz gráfica, la obtención de señales EEG y el clasificador también.

Existe una librería que se desarrolló para MATLAB, la cual permite accesar todos los canales del Emotiv EPOC. Es importante mencionar que para poder utilizar esta librería se debe contar con los archivos EDK proporcionados por Emotiv *Systems* al momento de comprar la licencia; sin estos no funciona la librería [29]. Se descargó la librería encontrada y se trató de graficar los canales en MATLAB para determinar que funcionara correctamente.

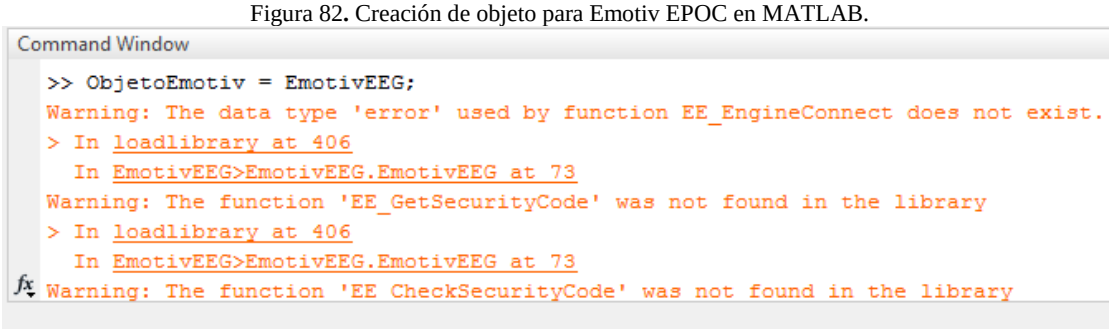
Existen tres configuraciones del Emotiv EPOC que se debían modificar según los requisitos de la aplicación de teclado de P300. La primera configuración que se modificó fue la frecuencia de muestreo del Emotiv, la cual se configuró a 128 Hz por canal. La segunda configuración es el tamaño del buffer donde se almacenarán temporalmente los datos obtenidos del Emotiv EPOC. La tercera configuración es el tiempo para las interrupciones del temporizador del Emotiv. La forma en que se actualizan los datos en el buffer es mediante un temporizador. En cada interrupción se actualizan los datos guardados en el buffer con los nuevos que se muestrearon con el Emotiv EPOC. Este temporizador y la frecuencia de muestreo definen la cantidad de datos nuevos que habrá en cada interrupción, ya que con mayor tiempo entre interrupciones se obtienen más señales del Emotiv EPOC. Por ejemplo, al utilizar una frecuencia de 128 Hz, si el temporizador se interrumpe cada 250 ms, se obtienen 32 muestras por canal; mientras que si se interrumpe cada 31.25 ms, se obtienen 4 muestras por canal.

Es importante mencionar que el buffer debe ser capaz de guardar, como mínimo, la cantidad de datos muestreados entre interrupciones. Por lo tanto, si el buffer solamente permite guardar 32 datos por canal, no se podría configurar la interrupción del temporizador a más de 250 ms porque se perderían muestras.

Como primer paso para implementar el Emotiv EPOC con MATLAB, se debe crear un objeto, el cual permitirá accesar toda la información que provee el EPOC y también permitirá modificar las configuraciones

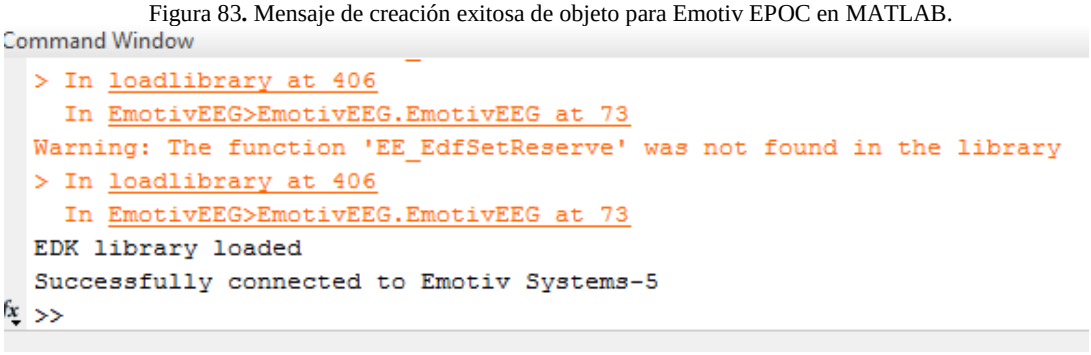
mencionadas anteriormente. No importa el nombre que se le coloca a este objeto, solamente es otra variable dentro del programa. La siguiente figura muestra cómo crear el objeto de EmotivEEG.

Figura 82. Creación de objeto para Emotiv EPOC en MATLAB.



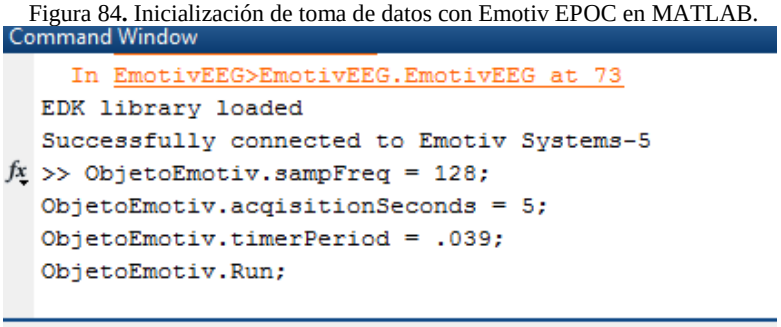
Aparecen advertencias en la ventana de MATLAB pero estas se refieren a funciones específicas del Emotiv que no se implementaron, tales como detección de parpadeo, verificación de conexión segura, entre otros. La Figura 83 muestra el mensaje que aparece cuando se haya creado exitosamente el objeto.

Figura 83. Mensaje de creación exitosa de objeto para Emotiv EPOC en MATLAB.



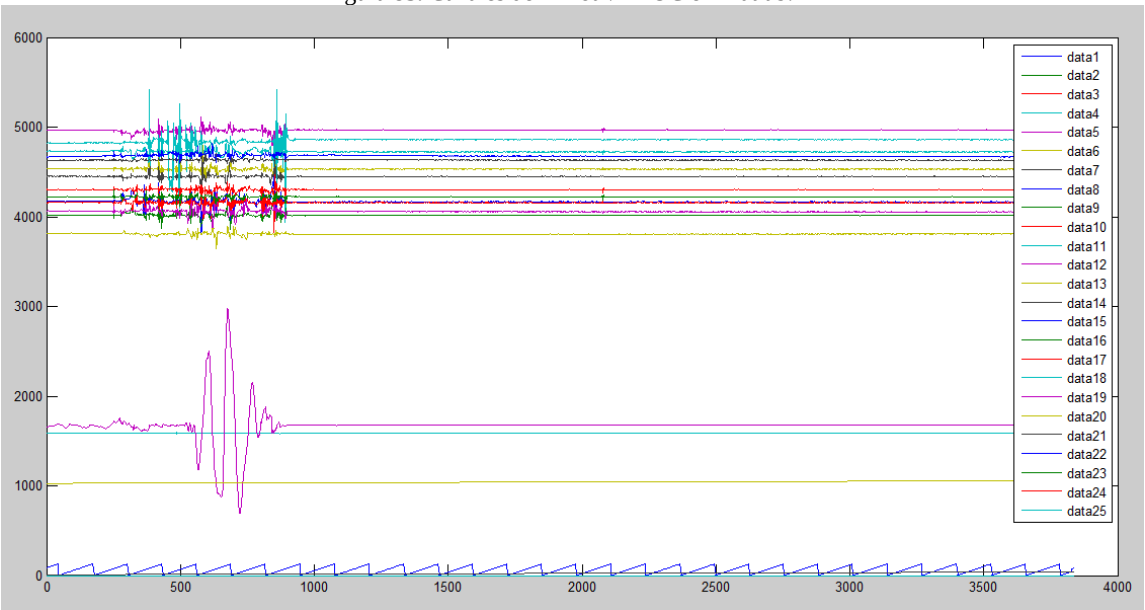
Una vez creado el objeto se puede modificar la frecuencia de muestreo, el tamaño del buffer y el tiempo de interrupción para el temporizador del Emotiv. Cuando se haya terminado de configurar el Emotiv se puede inicializar la toma de datos, como se muestra en la siguiente figura:

Figura 84. Inicialización de toma de datos con Emotiv EPOC en MATLAB.



Mientras corre el programa se pueden graficar los datos que se están obteniendo en cada canal, se estaría mostrando cinco segundos de información del Emotiv EPOC. En la Figura 85 se muestra los datos obtenidos en cada canal durante una grabación. La aplicación de teclado de P300 no requiere graficar los datos obtenidos del Emotiv, esta gráfica se realizó para verificar que se podía usar la información proporcionado por la librería.

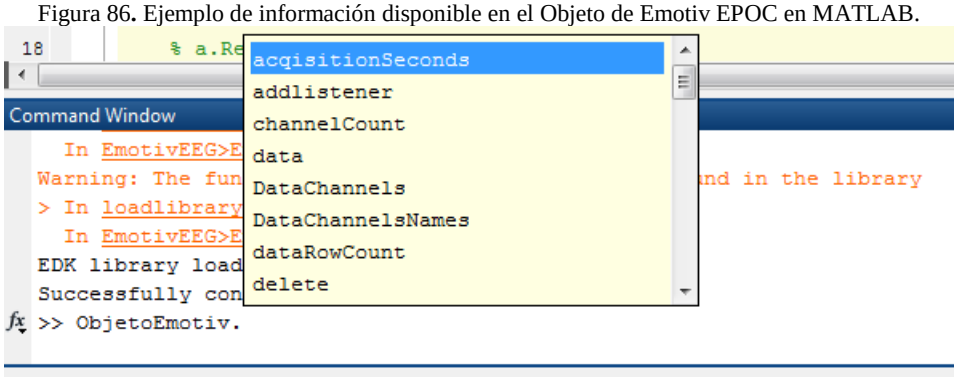
Figura 85. Canales del Emotiv EPOC en Matlab.



Dentro de la misma librería se indica qué representa cada canal. La figura anterior se creó cuando se giró el Emotiv sobre su eje y, por lo que la gráfica morada (Data 19) muestra los resultados del giroscopio en ese eje.

Dentro de la información que proporciona el EPOC, se encuentra una lista de los nombres de los canales, los datos de cada canal, información del temporizador, la frecuencia de muestreo, entre otros. También se puede utilizar cualquiera de las funciones que posee la librería, estas incluyen: Graficar los datos del Emotiv, actualizar una gráfica ya existente, borrar la graficar, guardar la gráfica, entre otros [29]. Para accesar a cualquiera de estos datos, se debe utilizar el objeto creado, escribiendo el nombre del objeto seguido de un punto, después se selecciona la información que se desea como se muestra en la Figura 86.

Figura 86. Ejemplo de información disponible en el Objeto de Emotiv EPOC en MATLAB.



c. Discusión. Para la aplicación de P300 es necesario filtrar la señal sin alterarla, en específico se debe evitar atenuar la señal. Esto se debe a que las amplitudes del P300 son muy pequeñas y cualquier atenuación podría eliminar por completo los ERPs. Al observar las respuestas de los filtros Butterworth y Chebyshev, se determinó que el filtro que mejor se adapta a la aplicación es el filtro Butterworth. Esto se debe a que mantiene una ganancia constante en toda su banda de paso, por lo que no atenuaría señales dentro de esas frecuencias. En cambio, un filtro Chebyshev posee una ganancia variable, a veces menor a 1, por lo que podría eliminar el ERP generado.

Debido a que la región de transición de los filtros Butterworth es mayor en comparación a los filtros Chebyshev, es posible que no todas las frecuencias en la banda de rechazo se eliminen. Una forma de minimizar la región de transición es aumentar el orden del filtro. Para ello se debe conectar en cascada dos filtros. La banda de paso no se ve afectado, solamente se atenúan aún más las frecuencias en la banda de rechazo. Por ello se construyeron dos filtros Butterworth de 6 polos y se conectaron en serie para tener un filtro Butterworth equivalente de 12 polos, como se muestra en la Figura 77.

Con los filtros analógicos diseñados se logró limpiar las señales EEG para poder observarlas en la computadora y tratar de determinar algún patrón en el lóbulo temporal y occipital. Al realizar las pruebas con el canal O1, no se logró determinar algún patrón. Se repitió la misma prueba varias veces y casi nunca se obtenía una gráfica igual a la anterior o a cualquiera de las otras. Esto se debe a que la amplitud del ERP, tales como el P300, es mucho menor a las señales EEG captados, por lo que no se logra observar a simple vista [89].

Como se puede observar en la Figura 79 y Figura 80, sí hubo cambio en las señales EEG del canal T3 ante diferentes estímulos auditivos. Sin embargo, la señal aún presenta mucho ruido. Posteriormente se verificó que se inducía más ruido con el simple hecho del acercamiento de otra persona a la cabeza del usuario. Por ende, el gorro de electrodos de *Electro-Cap International* es muy susceptible a ruido y alteraría demasiado las señales EEG, lo que dificultaría el entrenamiento del clasificador en el módulo de procesamiento y clasificación de señales [12].

Otra desventaja del gorro es que las señales varían bastante al quitar el gorro y volver a colocarlo. Por lo tanto, con cada sesión las señales serían distintas a las tomadas previamente. Como consecuencia, las señales EEG no serían consistentes con las captadas anteriormente y no se lograría obtener resultados aceptables.

Se demostró que el Emotiv EPOC logra atenuar el ruido en las señales EEG y que son de una mejor calidad en comparación con las señales obtenidas aplicando el filtrado analógico al utilizar el gorro de electrodos de *Electro-Cap International*. A partir de estos resultados se decidió utilizar el Emotiv para desarrollar la aplicación de deletreo de P300. Existe una librería de MATLAB para el Emotiv por lo que es posible accesar los canales del EPOC desde un mismo programa.

2. Desarrollo de sistema GUI con señales EEG

a. Diseño experimental.

Después de asegurar que se podía accesar los canales del Emotiv EPOC desde MATLAB, se continuó creando la interfaz gráfica de la aplicación de teclado de P300. Esto incluye la pantalla con la matriz de caracteres y la ventana de comandos para modo de operación *Online* y *Offline*, las cuales se explicarán en la siguiente sección.

Posteriormente se investigó sobre interrupciones en MATLAB para poder modificar la matriz e iluminar las filas y columnas en determinado momento. Después se realizaron pruebas utilizando la librería para el Emotiv para verificar que éste no generara errores.

Al verificar que funcionara la librería del Emotiv se realizó el modo de operación *Offline*. Se determinaron los tiempos de iluminación de las filas y columnas y se aseguró que estas fueran igual a las esperadas. Se realizaron las modificaciones necesarias para obtener los mejores tiempos posibles y que éstos se mantuvieran constantes durante toda la secuencia.

El modo de operación *Online* se empezó a desarrollar una vez verificado que el modo *Offline* estuviera funcionando adecuadamente. Para esta etapa era necesario incluir el código para el entrenamiento del clasificador en el módulo de procesamiento y clasificación de señales [12].

b. Resultados

1) Desarrollo aplicación de teclado P300.

Como primer paso para desarrollar la aplicación para P300 se realizó la interfaz gráfica del programa. Para esto se debía realizar una pantalla para la matriz de 6x6 caracteres, como se muestra en la Figura 87. En la parte superior izquierda se muestra el texto que se desea escribir y a la derecha, la letra actual que debería estar viendo la persona.

Figura 87. Ventana con matriz de caracteres.

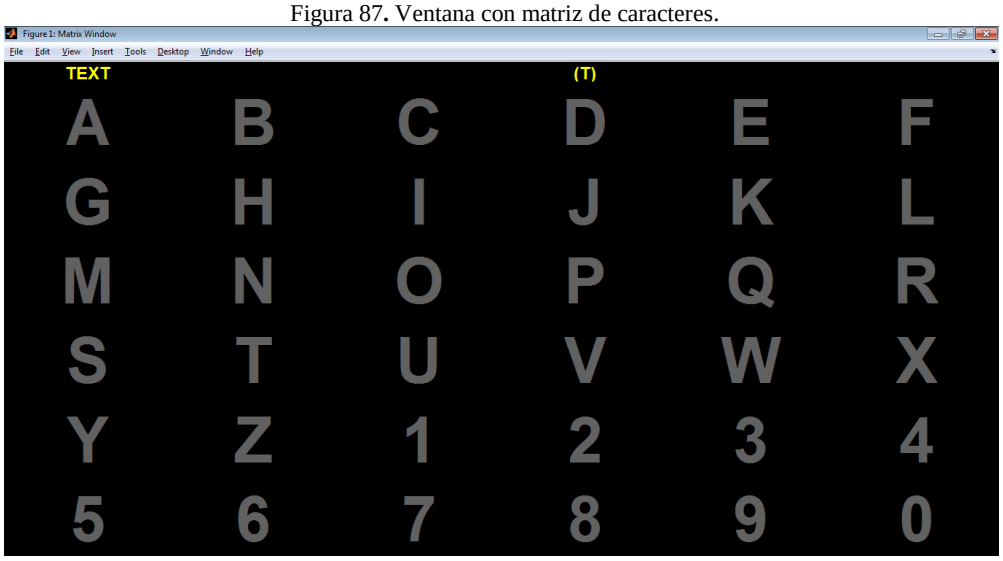
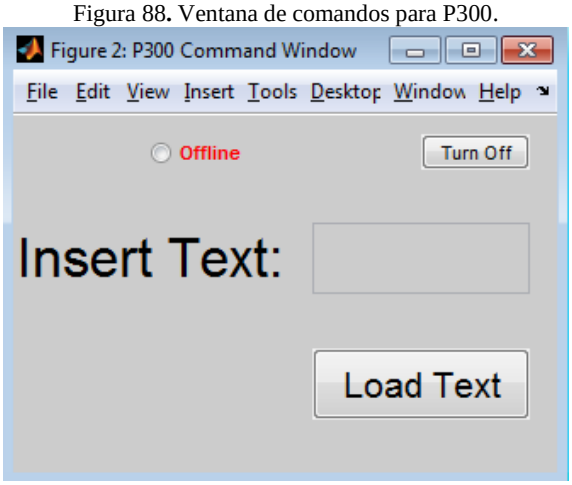


Figura 88. Ventana de comandos para P300.

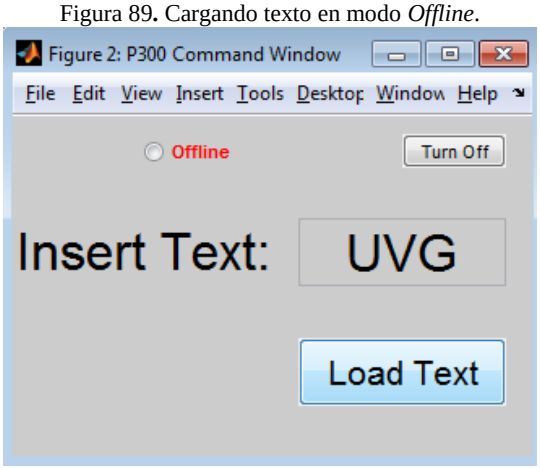


El programa de P300 consiste en iluminar las filas y columnas de la matriz de la Figura 87 de forma aleatoria. Se ilumina una fila o columna a la vez y no se vuelven a iluminar hasta que se hayan iluminado todas las filas y columnas. Al final del programa, cada fila y columna se habrá iluminado 15 veces, por lo tanto se contaría 30 iluminaciones por letra en la matriz. Esta secuencia se repite para todas cada letra en la palabra que se estará escribiendo. Por ejemplo, si desea escribir BET, la secuencia se haría para la letra B, después la E y de último la T, donde el programa finalizaría y se tendría que grabar la información obtenida de las señales EEG.

Se desarrollaron dos modos de operación del programa de P300, el modo *offline* y el modo *online*. Los dos modos son iguales en cuanto a cómo realizan las iluminaciones, las interrupciones utilizadas y la información que guarda, las diferencias se encuentran en cómo iniciar el programa de P300.

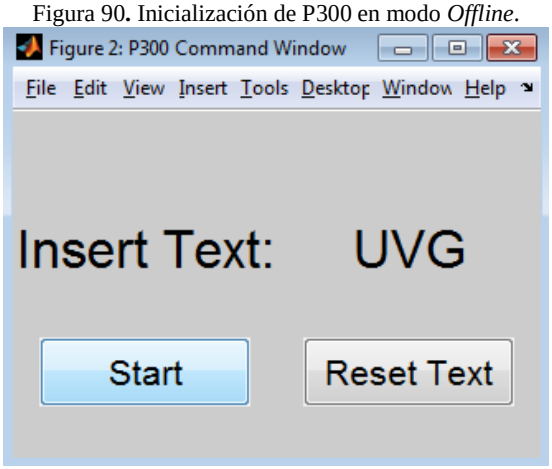
a) Modo de operación *Offline*. Se utiliza cuando se desea grabar corridas para entrenar el clasificador elaborado por [12]. Como paso inicial de este modo se debe escribir el texto que se desea deletrear y posteriormente cargar el texto presionando Load Text (Figura 89) Al hacer esto aparece un botón que dice Start, el cual sirve para empezar a iluminar la matriz (Figura 90).

Figura 89. Cargando texto en modo *Offline*.



Una vez terminada la secuencia se debe presionar *Reset Text*. Esto hace que se regrese a la pantalla que se observa en la Figura 89. Después se debe presionar *Turn Off* para desconectarse del Emotiv y poder grabar la información recopilada en la corrida.

Figura 90. Inicialización de P300 en modo *Offline*.



Para poder entrenar el clasificador se requiere que el programa de teclado de P300 guarde ciertos datos. Primero se necesitan las señales EEG en toda la corrida, como se mencionó anteriormente la frecuencia de muestreo es de 128 Hz. También es necesario saber qué fila o columna estaba prendida y en qué momento, además de saber cuándo se prendió la letra de interés, es decir, la letra que está viendo el usuario en ese

momento. Todos estos datos van identificados con un numero de muestra, el cual indica en qué muestra ocurrió el evento.

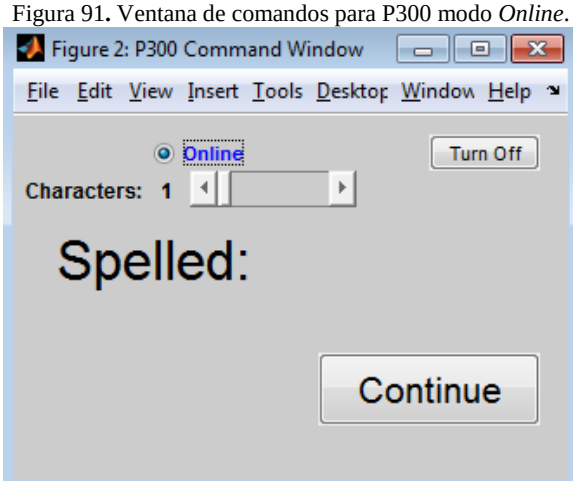
Para el entrenamiento del clasificador se requieren de tres sesiones del programa de P300. Cada sesión consiste en cinco palabras de tres letras cada uno. Para el proceso completo de entrenamiento se realizarían 15 palabras. Dos de las tres sesiones se utilizan para entrenar y la tercera sesión sirve para determinar qué tan bueno es el clasificador, en otras palabras, se utiliza para realizar pruebas y ver a cuántas letras le acertó con exactitud el clasificador.

b) Modo de operación *Online*. Se utiliza para deletrear en tiempo real, a comparación del modo *Offline*, en este modo sí se obtiene una predicción al finalizar la letra. Por lo tanto, como primer requisito se requiere un clasificador que ya se haya entrenado para ese usuario en específico. El código para entrenar el clasificador fue desarrollado por [12].

La reacción de las personas ante los estímulos externos es diferente para cada uno; sí es posible que el cerebro de dos personas reaccione de manera similar pero sí habría diferencias. Por ello es necesario utilizar un clasificador por persona y hacer el proceso completo de entrenamiento para cada uno.

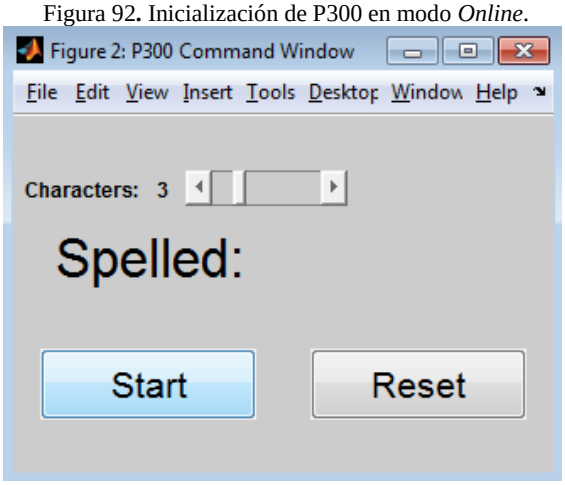
Para habilitar el modo online se debe presionar el botón que dice *Online* en la Figura 88. Al hacer esto se modificará la ventana de comandos a la que aparece en la siguiente figura:

Figura 91. Ventana de comandos para P300 modo *Online*.



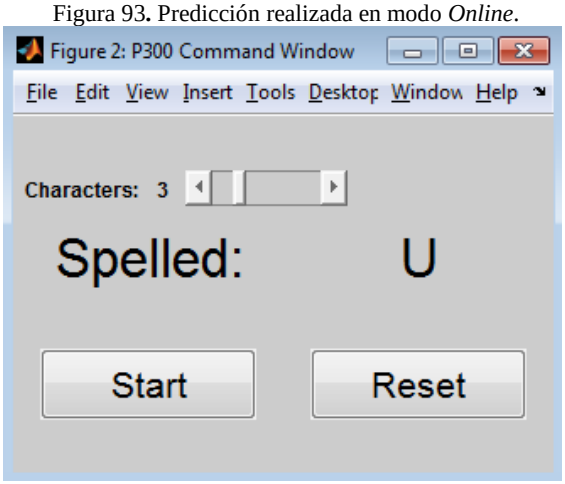
En este modo de operación solamente se debe elegir cuántas letras desea escribir el usuario, ya no se escribe el texto como se hacía en el modo *offline*. Posteriormente se presiona el botón de continuar el cual carga las configuraciones necesarias en la ventana con la matriz de caracteres y presenta la Figura 92. En la parte superior de la ventana con los caracteres se observará un número indicando qué número de letra de la palabra debería estar viendo la persona.

Figura 92**.** Inicialización de P300 en modo *Online*.



Al presionar *Start* se realizan las iluminaciones de manera normal, al terminar la secuencia para cada letra, se realiza la predicción de qué letra estaba viendo el usuario y aparece a la par de *Spelled* como se muestra en la Figura 93.

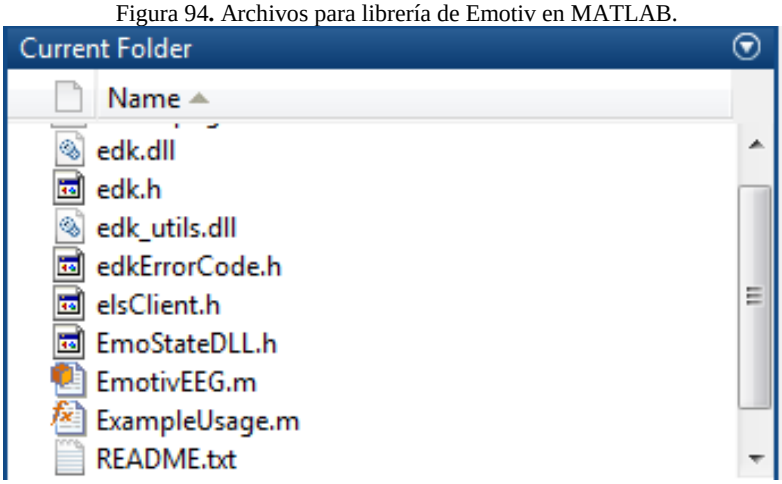
Figura 93**.** Predicción realizada en modo *Online*.



La secuencia se repite la cantidad de letras que se habían especificado. En el ejemplo anterior se repetiría tres veces y como resultado final se tendría tres predicciones. Para apagar el programa se debe presionar *Reset* y después *Turn Off*.

2) Implementación de herramientas de MATLAB para Emotiv. Una vez descargada la librería de Emotiv para MATLAB, se debe guardar en la carpeta de *Toolbox* de MATLAB. Dentro de la librería vienen dos archivos de MATLAB, EmotivEEG.m y ExampleUsage.m, aparte se debe de agregar los archivos EDK proporcionados por Emotiv con la compra de la licencia, como se muestra en la Figura 94. Se requiere de MATLAB de 32 bits para que esta librería funcione.

Figura 94. Archivos para librería de Emotiv en MATLAB.



EmotivEEG.m se utiliza para crear un objeto, el cual sirve para accesar a toda la información del Emotiv EPOC, esto incluye señales EEG, información de giroscopios, frecuencia de muestreo, etc. En la sección de anexos se encuentra el código utilizado para crear el objeto y obtener información de los canales.

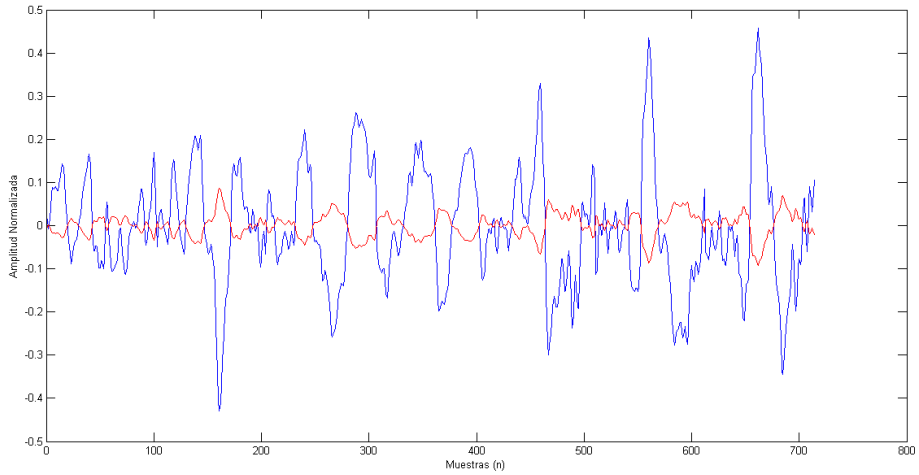
Como se observó en la Figura 85, el Emotiv posee 25 canales, pero solamente 14 de ellos corresponden a señales EEG. Los canales 4 a 16 son las señales EEG y están en el siguiente orden: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4, donde AF3 es el canal 4 y AF4 es el canal 16 [23]. Por lo tanto es necesario utilizar como mínimo estos canales, se pueden utilizar otros canales que contenga otra información si la aplicación así lo requiere. Para el teclado de P300 solamente se utilizaron las señales EEG.

3) Comparación y modificación de tiempos de iluminación en programa de teclado de P300.

Inicialmente el tiempo de iluminación de cada fila o columna era de 100 ms y el tiempo entre iluminaciones era de 75 ms. Al terminar la secuencia de una letra, existe un tiempo muerto de 1.5 s para que la persona pueda fijar su mirada en la siguiente letra que desea escribir. Para las iluminaciones se utilizó un temporizador de MATLAB, el cual permite interrupciones con una exactitud de hasta 1 ms.

La siguiente figura muestra la respuesta de un usuario ante los estímulos externos presentados, la función azul es la respuesta a estímulos de interés y la función roja es la respuesta a los demás estímulos.

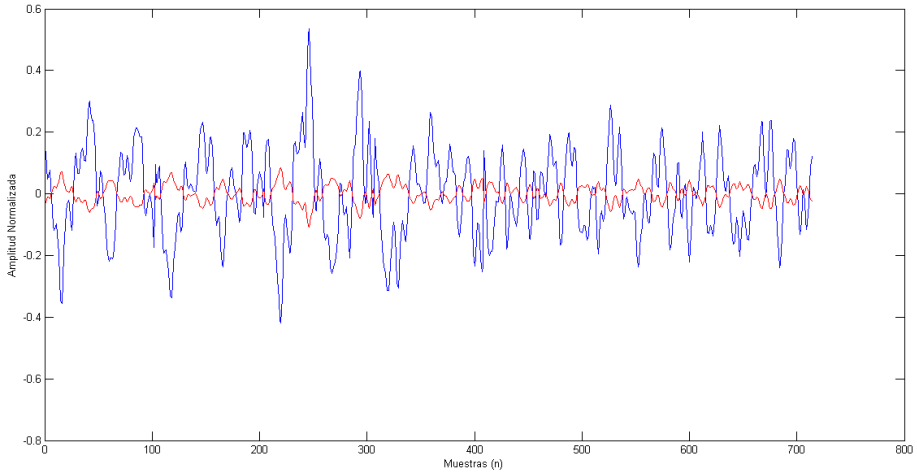
Figura 95**.** Respuestas positivas y negativas del usuario con tiempo de iluminación de 100 ms.



La librería de MATLAB para el Emotiv EPOC también utiliza un temporizador para interrumpir el programa y grabar los datos muestreados. Este temporizador se puede configurar, al igual que la frecuencia de muestreo del Emotiv. El temporizador se configuró para interrumpirse cada 39 ms y la frecuencia de muestreo del Emotiv se configuró a 128 Hz. Por lo tanto se recopilan 5 muestreas para cada canal entre interrupciones, es decir, se obtienen datos del Emotiv en bloques de 5 muestras por canal.

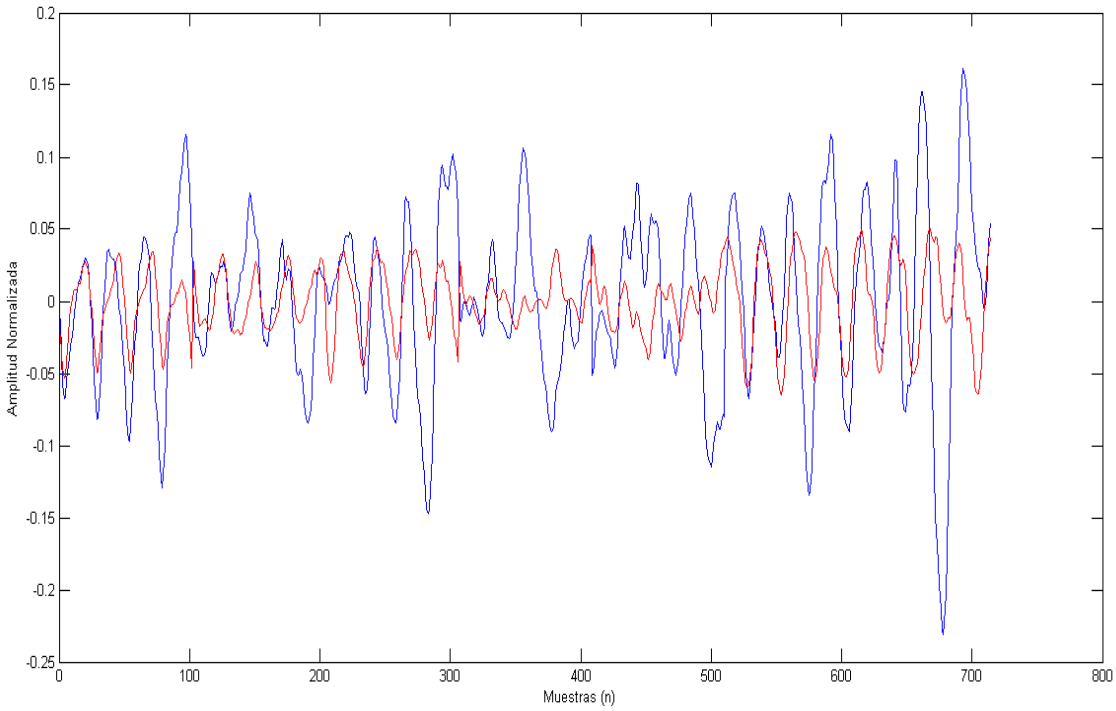
Se modificaron los tiempos de iluminación a 117 ms y el tiempo entre iluminaciones a 78 ms, para que estos tiempos fueran múltiplos del temporizado utilizado para el Emotiv EPOC. En la Figura 96 se muestra la respuesta del usuario ante los estímulos de interés y los demás estímulos. Posee la misma información que la figura anterior, sin embargo se desea determinar con qué tiempos de iluminación se obtiene mejor respuesta.

Figura 96**.** Respuestas positivas y negativas del usuario con tiempo de iluminación de 117 ms.



Se utilizó otra computadora con mejor procesador y más memoria RAM para determinar si existía variación en los resultados obtenidos. La Figura 96 muestra la respuesta del usuario ante los mismo estímulos externos pero utilizando otra computadora.

Figura 97**.** Respuestas positivas y negativas del usuario en otra computadora, con tiempo de iluminación de 117 ms.



Para las tres versiones anteriores se determinó la variación en los tiempos de iluminación y entre iluminaciones para verificar si los estímulos poseían un tiempo de duración constante. Para determinar esto se obtuvo la duración de cada iluminación y el tiempo entre todas las iluminaciones y se comparó con el tiempo deseado. Se encontró cuántas eran diferentes a las deseadas y se representó este valor como un porcentaje de todas las iluminaciones que ocurrieron. Por ejemplo, si había en total 180 iluminaciones pero 20 tuvieron una duración diferente a la deseada, entonces esa corrida tuvo un 9% de variación en sus tiempos de iluminación. Se realizó lo mismo para los tiempos entre iluminaciones.

Cuadro 12**.** Variaciones en tiempos de iluminaciones.

Procesador de Computadora	Periodo Timer Emotiv (ms)	Variación Iluminaciones	Variación entre iluminaciones
AMD Athlon Dual-Core, 2 GB de RAM	250	37.24%	18.88%
AMD Athlon Dual-Core, 2 GB de RAM	39	16.91%	17.17%
Intel Core (I5), 4 GB de RAM	39	20.86%	15.10%

También se consideró como factor, la variación entre sesiones. Los resultados en el cuadro anterior muestran la variación de las iluminaciones en una sola corrida. Este porcentaje puede cambiar para cada

corrida, pero lo ideal sería que todas las corridas tengan la misma variación. Es decir, no se desea que en una corrida la variación sea de 4% y en la siguiente se tenga una variación de 40%. Se realizaron 15 sesiones con la computadora con procesador AMD Athlon y 15 con la computadora Intel Core I5. Para ambas se utilizó un temporizador de 39 ms para el Emotiv.

Cuadro 13. Variación en iluminaciones para tres sesiones utilizando computadora con procesador AMD Athlon.

Sesión	Corrida	Variación Iluminaciones	Variación entre iluminaciones
1	1	6.85%	5.56%
	2	7.41%	5.56%
	3	9.07%	10.37%
	4	9.26%	6.11%
	5	8.33%	6.85%
2	1	8.33%	9.07%
	2	20.74%	22.41%
	3	7.96%	9.26%
	4	29.81%	29.81%
	5	12.59%	10.37%
3	1	55.19%	59.63%
	2	6.48%	6.85%
	3	7.41%	7.22%
	4	11.85%	12.04%
	5	52.41%	56.48%
Variación Máxima		55.19%	59.63%
Variación Mínima		6.48%	5.56%
Promedio		16.91%	17.17%
Desviación Estándar		16.23%	17.90%

Cuadro 14. Variación en iluminaciones para tres sesiones utilizando computadora con procesador Intel I5.

Sesión	Corrida	Variación Iluminaciones	Variación entre iluminaciones
1	1	20.74%	15.93%
	2	21.48%	15.37%
	3	21.11%	14.81%
	4	22.78%	13.70%
	5	21.48%	13.70%
2	1	17.78%	16.85%
	2	19.07%	16.11%
	3	22.41%	15.74%
	4	20.93%	13.89%
	5	19.81%	15.56%
3	1	20.19%	16.48%
	2	21.48%	13.89%
	3	21.48%	14.44%
	4	20.93%	16.48%
	5	21.30%	13.52%
Variación Máxima		22.78%	16.85%
Variación Mínima		17.78%	13.52%
Promedio		20.86%	15.10%
Desviación Estándar		1.26%	1.17%

c. Discusión. Algunos factores que se tomaron en cuenta para la interfaz gráfica del P300 es el contraste en los colores utilizados para el estímulo visual. Las letras se volvían blancas cuando se iluminaba la fila o columna de la matriz. Normalmente las letras estaban de color gris cuando no estaban siendo iluminadas, es importante mencionar que un cambio más llamativo para el usuario sería que las letras fueran casi negras y cuando se iluminaran, fueran blancas completamente. Sin embargo, no es deseable un contraste tan fuerte ya que existen efectos negativos también. El usuario debía tratar de ignorar las iluminaciones de las otras letras, pero con un contraste muy fuerte, la iluminación de filas adyacentes al que está viendo el usuario se vuelven más llamativos, provocando un P300 en el cerebro. Por lo tanto se llegaría a provocar P300 para otras letras que el usuario no está viendo, esto disminuiría la exactitud del teclado.

Otra opción que se analizó para que el estímulo fuera más llamativo, era aumentar el tamaño de las letras que se iluminaban y posteriormente regresarlos a su tamaño normal. Se determinó que este cambió sí era más llamativo para el usuario y aún lograba ignorar las iluminaciones de las filas y columnas adyacentes. Esto fue un resultado positivo porque aumentó la respuesta a los estímulos de interés y no afectó la respuesta para los demás estímulos.

Las pruebas iniciales que se realizaron con el teclado de P300 poseían un tiempo de iluminación de 100 ms y un tiempo entre iluminaciones de 75 ms. Los resultados en la siguiente sección describen el desempeño del programa y muestran que existe mucho error en las predicciones realizadas. Para entender mejor cuál era el problema se empezó a graficar la respuesta del usuario ante los estímulos de interés y los demás estímulos, la cual se muestra en la Figura 95 [12]. También se determinó la variación en los tiempos de estimulación durante las corridas, es decir, se determinaba si todas las estimulaciones duraron el mismo tiempo o no.

Según el módulo de procesamiento y clasificación de señales, la función en la Figura 95 debería ser más suave, es decir, no debería presentar tantos picos en su amplitud. En el Cuadro 12 se muestra el porcentaje de variación en el tiempo de las iluminaciones (37.24%) y entre iluminaciones (18.88%). Esto quiere decir que 37.24% las iluminaciones en la corrida no duraron lo esperado. Esto daba la idea que el problema era de sincronización entre el muestreo y las iluminaciones.

Para referenciar todas las señales del teclado de P300 se utilizó el número de muestra no el tiempo transcurrido. Para determinar en qué muestra ocurre cada evento se tomó el tiempo desde que inicia el programa hasta el momento donde ocurre el evento y después se multiplica este valor por la frecuencia del Emotiv (128 Hz). Por ejemplo, si la primera iluminación ocurrió medio segundo después de haberse iniciado el programa, la iluminación ocurrió en la muestra 64 y las muestras utilizadas para el clasificador serían referenciadas a partir de esta muestra. Por lo tanto, si existe un error en el cálculo de la muestra donde ocurrió la iluminación y realmente ocurrió muestras antes de lo calculado, se estarían obteniendo señales EEG que no corresponden a ese estímulo.

Debido a que la frecuencia del Emotiv es de 128 Hz, el tiempo entre muestras es de 7.8125 ms. Los tiempos de 100 ms y 75 ms utilizados para los tiempos de iluminación y entre iluminaciones, respectivamente, no son múltiplos de 7.8125 ms. En cantidad de muestras, 75 ms equivale a 9.6 muestras y 100ms equivale a 12.8 muestras. Se requieren de cantidades enteras de muestras para poder referenciar la iluminación y redondear estos valores no es la mejor solución. Al redondear la muestra donde ocurre el estímulo, se mueve la referencia que se utiliza para la toma de señales EEG, ya sea una muestra antes o después del número real de la muestra. Lo ideal sería no tener que redondear el número de muestra, por lo que se debe procurar utilizar temporizadores con duraciones de tiempo que sean múltiplos de 7.8125 ms. Esto resultaría en menos error al redondear las muestras, lo que implica que las señales EEG tomadas serían las correctas.

Por ello se decidió utilizar un temporizador de 39 ms para el Emotiv (esto equivale a 5 muestras). Es importante mencionar que para que el sistema se mantenga sincronizado, las interrupciones de iluminaciones debe ser un múltiplo de las interrupciones de muestreo. Para el tiempo entre iluminaciones se utilizó 78 ms y para las iluminaciones se utilizó 117 ms. Al hacer esto se asegura que las iluminaciones no se desfasarán

con respecto a las muestras tomadas. Si no se cumple esto, es posible que el tiempo entre la interrupción para una iluminación y la interrupción de muestreo, que ocurre inmediatamente después de la iluminación, varíe con cada nueva iluminación.

Se probaron estos cambios en la computadora con procesador AMD Athlon y 2GB de RAM para determinar si los resultados mejoraron. Se observó mejora pero aún no se lograba resultados tan buenos como en otros programas de teclado de P300 mostrados en el módulo de procesamiento y clasificación de señales [12]. La Figura 96 muestra la respuesta del usuario obtenido al utilizar este nuevo temporizados. Se observa que la cantidad de cambios bruscos en la señal disminuyeron, sin embargo, aún existen algunas secciones donde se corta la señal. El Cuadro 12 muestra que la variación de la duración de los estímulos disminuyó en comparación con los obtenidos utilizando el temporizador de 250 ms para el Emotiv. Esto hace pensar que el problema podría ser la velocidad de procesamiento de la computadora, lo que afecta la capacidad de captar todas las señales EEG correctamente. Por ello se decidió utilizar otra computadora con mejor procesador y mayor cantidad de memoria RAM.

La Figura 96 muestra la respuesta del usuario obtenido con el mismo temporizador que antes, pero corriendo el programa en otra computadora. La función obtenida es muy distinta al anterior y casi no posee cambios bruscos en su amplitud. En otras palabras, parece ser más continua que las de la Figura 95 y Figura 96. En cuanto a la variación de sus iluminaciones, tiende a ser igual que los obtenidos con la computadora con procesador AMD Athlon. Sin embargo, el Cuadro 13 y Cuadro 14 muestra las variaciones en iluminaciones para ambos casos. Aparte de tener poca variación en los tiempos de iluminación, se requiere que todas las corridas sean parecidas. El Cuadro 13 muestra las variaciones para 15 corridas, en promedio se determinó que hubo una variación de 16.91%, sin embargo, habían corridas que poseían variaciones de 55.19% y otras con 6.48%. El Cuadro 14 muestra que las variaciones estuvieron en un rango entre 17.78% y 22.78%. Posteriormente se obtuvo la desviación estándar para ambos casos para determinar cuál poseía mayor varianza. Se comprobó que se producía una menor variación en los estímulos utilizando la computadora con procesador Intel I5 y 4GB de RAM.

Con respecto a la librería disponible para controlar el Emotiv con MATLAB, se requiere tener MATLAB de 32 bits instalado. Dicha librería no funciona si se utiliza Matlab de 64 bits porque los archivos edk.h y edkErrorCode.h utilizan datos de información de 32bits y se producen errores al utilizar esa versión de MATLAB.

3. Evaluación de desempeño de teclado P300

a. Diseño experimental. Se realizaron pruebas con diferentes personas para tratar de determinar el rendimiento del teclado P300. Las pruebas consistían en tres sesiones de cinco corridas cada una. Cada corrida consiste en una palabra de tres letras, por lo tanto se utilizaron 15 letras por corrida.

Se grababan las tres sesiones en modo *offline*, posteriormente se utilizaría el código del módulo de procesamiento para entrenar el clasificador [12]. Para el entrenamiento solamente se utilizaba las dos primeras sesiones tomadas, la tercera sesión se utilizaba para determinar el desempeño del clasificador y del teclado P300. De esta última sesión, se evaluaba cuántas letras correctas había predicho el clasificador con las señales obtenidas del programa de teclado P300 desarrollado. Esto permitiría verificar que el programa de teclado P300 fue desarrollado apropiadamente, es decir, que obtenía las señales EEG de manera correcta y que los estímulos visuales estuvieran sincronizados con las muestras tomadas.

b. Resultados. Como se describió en la sección anterior, se realizaron cambios en los temporizadores que se utilizaron para el programa de teclado de P300. Para la versión que utilizó un temporizador de 250 ms, se observó que la respuesta obtenida no era la correcta (Figura 95). Se realizaron las modificaciones en el temporizador y se intentó utilizar el teclado de P300. Cuadro 15 y Cuadro 16 muestran las predicciones generadas con esta versión del programa.

Cuadro 15. Primera sesión utilizando el teclado de P300 con un temporizador de 39 ms.

Corrida	Palabra	Predicción
JC2006S002R01V4	JAV	JOV
JC2006S002R02V4	KTM	KTT
JC2006S002R03V5	DEN	AWL
JC2006S002R04V5	3SJ	ZSC
JC2006S002R05V6	TAX	BA0
	Exactitud	40.00%

Cuadro 16. Segunda sesión utilizando el teclado de P300 con un temporizador de 39 ms.

Corrida	Palabra	Predicción
JC2006S003R01	W7N	47J
JC2006S003R02	BOX	HOX
JC2006S003R03	M6Q	M6Q
JC2006S003R04	8AE	0E3
JC2006S003R05	RO5	R7I
	Exactitud	46.67%

La exactitud en las dos sesiones tomadas no es la deseada, por lo que se decidió utilizar otra computadora con mejor procesador y RAM.

Cuadro 17. Primera sesión de teclado de P300 con computadora con procesador I5 y temporizador de 39 ms.

Corrida	Palabra	Predicción
JC2406S002R01	8AT	7AT
JC2406S002R02	7QM	5QM
JC2406S002R03	OF5	RF5
JC2406S002R04	WXP	WXH
JC2406S002R05	NID	NIC
	Exactitud	66.67%

Cuadro 18. Segunda sesión de teclado de P300 con computadora con procesador I5 y temporizador de 39 ms.

Corrida	Palabra	Predicción
JC2906S002R01	1VY	1V4
JC2906S002R02	OU7	OU7
JC2906S002R03	5EP	5EP
JC2906S002R04	3TO	3TO
JC2906S002R05	MA8	MA8
	Exactitud	93.33%

Se notó que los electrodos estaban oxidados, por lo que se pidieron más y se realizaron otras pruebas para comprobar el desempeño del sistema.

Cuadro 19. Primera sesión de teclado de P300 con la configuración anterior pero con nuevos electrodos.

Corrida	Palabra	Predicción
JC2308S001R01	IVA	ITG
JC2308S001R02	H0T	H0T
JC2308S001R03	TED	NEC
JC2308S001R04	3RT	3RK
JC2308S001R05	S15	U15
	Exactitud	60.00%

Cuadro 20. Segunda sesión de teclado de P300 con la configuración anterior pero con nuevos electrodos.

Corrida	Palabra	Predicción
JC2308S002R01	SE7	SED
JC2308S002R02	KID	KH8
JC2308S002R03	13B	L3B
JC2308S002R04	8NT	8NT
JC2308S002R05	A2K	A2K
	Exactitud	73.33%

c. Discusión. Al modificar el tiempo de interrupción del temporizador del Emotiv y de las iluminaciones, se logró obtener predicciones con el programa de teclado de P300. Los resultados no eran satisfactorios, por lo que se trató de determinar si aún había problemas de sincronización. Se infirió que el problema de sincronización siempre existía pero que el problema principal era la computadora que se estaba utilizando. Se utilizó otra computadora y se realizaron más pruebas. Comparando el Cuadro 17 y Cuadro 18 con el Cuadro 15 y Cuadro 16 se comprueba que la exactitud obtenida es mejor que antes. Se concluye que utilizar uno de los problemas del programa de teclado P300 era la computadora que estaba siendo utilizada.

Al realizar más pruebas se observó que los electrodos estaban oxidándose y que era difícil establecer una buena conexión en cada canal del Emotiv EPOC. Por ello es que las exactitudes en el Cuadro 17 y Cuadro 18 son tan distintas. Entre veces se logra obtener una buena conexión en todos los canales y otras veces no, lo que hace que las señales EEG obtenidas con el Emotiv en cada canal sean muy distintas entre corridas. Con esto no se lograría obtener un clasificador muy robusto porque las señales varían con cada sesión y se espera que las señales del cerebro sean similares para la mayoría de sesiones [12]. Se ordenaron más electrodos para seguir realizando pruebas y asegurar que los electrodos no eran el problema.

El Cuadro 19 y Cuadro 20 no llegan a tener exactitudes tan altas como en el Cuadro 17 y Cuadro 18, pero estos resultados fueron obtenidos utilizando un clasificador que se había obtenido hace un mes. Lo que implica que después de un mes, las señales EEG obtenidas por el programa de teclado de P300 aún se parecen a las grabadas inicialmente. Existen muchos factores que influyen en los resultados del teclado, entre ellos: el pelo, el estado de la persona, si se encuentra cansada o no, su humor, si se encuentra distraída durante el experimento o no, entre otros [26]. Por lo tanto, es necesario hacer pruebas con más usuarios para poder generalizar los resultados obtenidos y determinar el desempeño del programa. En la siguiente sección se discute los resultados obtenidos con varios sujetos de prueba y se compara el rendimiento del teclado P300 desarrollado en este trabajo con otros sistemas existentes.

C. INTEGRACIÓN DE MÓDULOS

Una vez ambos módulos fueron optimizados lo más posible en su funcionamiento individual, se procedió a integrarlos para tener un sistema completo funcional, como se mostrará a lo largo de esta sección.

1. Selección de cadena de procesamiento. En este caso, se aplicaron las dos cadenas de procesamiento que brindaron una mayor exactitud promedio dentro de las opciones analizadas al utilizar el módulo de procesamiento y clasificación en conjunto con el programa BCI2000 y el Emotiv como dispositivo de adquisición de datos. Dichas cadenas fueron B.5 y B.7. A pesar de que la cadena B.3 tuvo la misma exactitud promedio que la cadena B.5, esta no fue considerada por utilizar un filtro FIR cuyo orden era muy elevado como para ser usado en tiempo real. Estas cadenas fueron probadas sobre un conjunto de cuatro

individuos, todos ellos varones de 23 años. En este caso, se grabaron tres sesiones, cada una de cinco corridas de tres letras. De estas, dos fueron utilizadas para entrenar y validar a los clasificadores, mientras que la tercera fue usada para calcular la exactitud.

a. Cadena de procesamiento C.1. Esta cadena de procesamiento corresponde a la B.5, agregando el uso del ensamble de tres clasificadores:

- Se aplicó un filtro espacial CAR a las señales EEG.
- Se aplicó a las señales EEG un filtro Butterworth pasa-banda IIR de 1 a 30 Hz, de orden 6.
- Se aplicó una normalización espacial, por corrida, a unidades tipificadas z.
- Se tomaron todas las muestras, de todos los canales, de los 800 ms posteriores a la estimulación.
- Se aplicó una decimación con un factor de dos.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.
- Se entrenaron y validaron tres clasificadores, los cuales fueron utilizados para realizar tres clasificaciones individuales, así como también para realizar una sola clasificación usando las salidas de los tres.

Cuadro 21. Desempeño de la cadena de procesamiento C.1

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)	Exactitud con tres clasificadores (%)
B.01	0.0001	10	88	78	82	46.67	80.00
	0.001	10	75	67	70	66.67	
	0.01	3,000	75	67	70	88.67	
B.02	0.003	30	100	22	36	66.67	91.67
	0.003	30	100	67	80	83.33	
	0.0001	300	100	22	36	75.00	
B.03	0.0001	300	50	44	47	33.33	40.00
	0.001	30	100	33	50	46.67	
	0.001	100	50	56	53	26.67	
B.04	0.01	1,000	83	56	67	53.33	46.67
	0.001	10	100	33	50	53.33	
	0.003	30	100	44	62	46.67	
					Promedio	57.25	64.58

b. Cadena de procesamiento C.2. Esta cadena de procesamiento corresponde a la B.7:

- Se aplicó un filtro espacial CAR a las señales EEG.
- Se aplicó a las señales EEG un filtro pasa-banda IIR de 1 a 15 Hz, de orden 4.
- Se aplicó una normalización espacial, por corrida, a unidades tipificadas z.
- Se tomaron todas las muestras, de todos los canales, de los 800 ms posteriores a la estimulación.
- Se aplicó una decimación con un factor de dos.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.

- Se entrenaron y validaron tres clasificadores, los cuales fueron utilizados para realizar tres clasificaciones individuales, así como también para realizar una sola clasificación usando las salidas de los tres.

Los resultados de las dos cadenas anteriores se encuentran en el Cuadro 21 y en el Cuadro 22, respectivamente. A pesar de que la única diferencia entre ambas cadenas de procesamiento era el filtro utilizado, la cadena de procesamiento C.2 exhibió una mejor exactitud promedio, tanto para el caso de los clasificadores individuales como para el ensamble de clasificadores. La diferencia entre la exactitud promedio de los clasificadores individuales fue de 14.69% y, entre los ensambles de clasificadores, 11.67%, probando que la cadena de procesamiento C.2 es más adecuada que la C.1 para el procesamiento y clasificación de señales EEG relacionadas con un teclado P300. A pesar de esto, se propuso una modificación más, inspirada en la literatura [42], para mejorar el rendimiento obtenido, modificación que se presenta en la cadena de procesamiento C.3.

Cuadro 22. Desempeño de la cadena de procesamiento C.2

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)	Exactitud con tres clasificadores (%)
B.01	0.0001	30	83	56	67	80.00	80.00
	0.001	30	59	56	58	80.00	
	0.001	30	100	89	94	80.00	
B.02	0.001	100	50	22	31	83.33	91.67
	0.003	30	67	44	53	83.33	
	0.01	30	67	22	33	83.33	
B.03	0.003	30	100	33	50	46.67	60.00
	0.0001	300	100	22	36	53.33	
	0.01	30	100	11	20	60.00	
B.04	0.003	100	100	44	62	80.00	73.33
	0.01	30	71.43	55	63	66.67	
	0.003	30	75	33	46	66.67	
					Promedio	71.94	76.25

c. Cadena de procesamiento C.3.

- Se aplicó un filtro espacial CAR a las señales EEG.
- Se aplicó a las señales EEG un filtro pasa-banda IIR de 1 a 15 Hz, de orden 4.
- Se aplicó una normalización espacial, por corrida, a unidades tipificadas z.
- Se tomaron las muestras, de los canales F3, FC5, P7, O1 O2, P8, FC6 y F4, del tiempo transcurrido entre los 100 y 800 ms posteriores a la estimulación.
- Se aplicó una decimación con un factor de dos.
- Se promediaron las muestras correspondientes al mismo elemento de la matriz.
- Se entrenaron y validaron tres clasificadores, los cuales fueron utilizados para realizar tres clasificaciones individuales, así como también para realizar una sola clasificación usando las salidas de los tres.

En esta cadena, se tomó al procesamiento que brindó una mayor exactitud promedio dentro de las opciones anteriormente contempladas y se realizó una mayor selectividad respecto a las muestras a considerar: Para empezar, se excluyeron las muestras correspondientes a los 100 ms posteriores a la estimulación, debido a que el P300 es una señal con una latencia de 300 ms, por lo que antes de dicho tiempo, no se presentará. Para dejar un margen de seguridad, no se excluyeron los primeros 300 ms, sino únicamente los primeros 100 ms. Además, se tomaron en cuenta únicamente los canales que se hallaban ubicados sobre las regiones parietal, occipital y central del cerebro. La región parietal se eligió por ser, de acuerdo a la teoría, el área donde el P300 se manifiesta con mayor intensidad, mientras que las otras dos regiones fueron seleccionadas tomando como referencia los trabajos de Manyakov [48], Krusienski [44] y Kaper [42], quienes utilizaron las regiones occipital y central en sus estudios. El objetivo de esta selección de las muestras a considerar era el de disminuir la dimensión de los vectores a clasificar, además de retirar aquellas muestras que quizás solo introducían ruido a la clasificación, todo ello, con el objetivo de facilitar la tarea del algoritmo de aprendizaje automático.

Cuadro 23. Desempeño de la cadena de procesamiento C.3

Sujeto	C	σ	Precisión (%)	Exhaustividad (%)	Puntaje F_1 (%)	Exactitud (%)	Exactitud con tres clasificadores (%)
B.01	0.003	30	100	78	88	86.67	86.67
	0.0001	30	100	44	62	86.67	
	0.0001	100	80	89	84	80.00	
B.02	0.001	10	60	33	43	58.33	83.33
	0.003	30	50	33	40	91.67	
	10	30	50	11	18	83.33	
B.03	0.01	100	43	33	38	66.67	60.00
	0.0001	300	50	11	18	46.67	
	0.3	10,000	63	56	59	53.33	
B.04	0.003	10	75	67	70	66.67	66.67
	0.003	10	50	33	40	80.00	
	30	100	56	56	56	60.00	
					Promedio	71.67	74.17

Los resultados de la cadena de procesamiento C.3 se presentan en el Cuadro 23, donde se puede verificar que se obtuvo un desempeño similar, pero no mejor, al de la cadena C.2. Esto indica que parte de los datos retirados permitían que el clasificador distinguiera de mejor manera las dos clasificaciones posibles. Debido a su superior exactitud promedio, tanto para los clasificadores individuales como agrupados en ensamble, se decidió conservar la cadena de procesamiento C.2.

d. Comparación entre diferentes clasificadores y métodos de adquisición de señales. Con el objetivo de poder comparar el rendimiento del clasificador propuesto en este trabajo con clasificadores ya existentes de una manera justa, se realizó lo siguiente: Utilizando los mismos cuatro sujetos cuyos resultados fueron utilizados para comparar las cadenas implementadas en la sección anterior, se grabaron dos sesiones para cada sujeto, pero utilizando BCI2000. Los datos de dichas sesiones fueron utilizadas para entrenar al clasificador de ondas P300 incluido con BCI, utilizando para ello 8 corridas como sesiones de entrenamiento

y 2 de prueba. En este caso no se necesitan sesiones de validación, puesto que el algoritmo usado por este clasificador (SWLDA) no lo requiere. El clasificador de BCI2000 fue entrenado utilizando su configuración estándar, la cual corresponde a considerar un máximo de 60 características, con un valor p de entrada de 0.1 y uno de salida de 0.15. Además, se consideran los 800 ms posteriores a la estimulación en todos los canales disponibles, no se aplica filtro espacial alguno y se realiza una decimación a 20 Hz. Luego, utilizando el modelo de clasificación creado por BCI2000, se grabó una tercera sesión para cada sujeto utilizando BCI2000, pero en esta, se registraron las predicciones realizadas por el programa y se contrastaron con la palabra esperada.

Utilizando estos mismos datos, se entrenó el clasificador propuesto en este trabajo, usando las primeras dos sesiones como datos para entrenar, validar y probar al clasificador, y la tercera, para obtener su exactitud, usando un ensamble de tres SVMs. Las exactitudes obtenidas en estos dos casos fueron comparadas con las exactitudes obtenidas para el ensamble de SVMs en la cadena de procesamiento C.2, que utilizaba el módulo de obtención de señales. La comparación debió de ser realizada de esta manera puesto que, para que esta fuera justa, se necesitaba que los datos sobre los cuales se iban a realizar las pruebas fueran lo más parecidas posibles. Esto quiere decir que se debían de utilizar los mismos sujetos, el mismo dispositivo de adquisición de señales de EEG y, de ser posible, operar sobre una misma grabación (ya que en diferentes grabaciones, algunas variables, como la concentración del paciente y la ubicación de los electrodos, pueden fluctuar). Siguiendo el procedimiento propuesto, se garantizaban los primeros dos puntos. No obstante, el tercer punto (que los datos pertenecieran a la misma grabación) no podía aplicarse en el tercer conjunto de datos considerado (aquellos grabados utilizando el módulo de obtención de señales de este megaproyecto) ya que, a pesar de que tanto dichos datos como los obtenidos por BCI2000 utilizaron como dispositivo de adquisición de señales al Emotiv EPOC, cada uno utilizaba un programa distinto para estimular al usuario y almacenar los datos. No obstante, como ya se mencionó anteriormente, con las medidas tomadas se buscó hacer que las condiciones de prueba fueran lo más cercanas posibles. Además, es importante considerar que por estas mismas razones no se podían comparar los resultados de este trabajo directamente con otros resultados en la literatura, puesto que los sujetos de prueba y dispositivos de adquisición de datos eran otros (ver, e.g., el trabajo de Manyakov [48] y Krusienski [44]).

Cuadro 24. Comparación de la exactitud obtenida al usar diferentes clasificadores y métodos de adquisición de datos.

Sujeto	Exactitud promedio (%)		
	Grabación y clasificador de BCI2000	Grabación en BCI2000 y clasificador propuesto	Grabación y clasificador propuestos
B.01	60.00	100	80.00
B.02	46.67	86.67	91.67
B.03	46.67	66.67	60.00
B.04	40.00	46.67	73.33
Promedio	48.34	75.00	76.25

Los resultados de la comparación se presentan en el Cuadro 24. En dicho cuadro, la primera columna muestra la exactitud promedio obtenida utilizando tanto la grabación de señales EEG, como el clasificador, del programa BCI2000. En la segunda columna, se presenta el resultado de la grabación usando la grabación de BCI2000, pero el clasificador desarrollado en el módulo de procesamiento y clasificación de señales EEG, mientras que la última columna presenta el resultado obtenido al utilizar el sistema integrado propuesto en este trabajo. Al observar el cuadro, se puede notar claramente que el clasificador propuesto es superior al incluido con el programa BCI2000, ya que, aun operando sobre los mismos datos, el clasificador de este trabajo obtiene una exactitud que es 1.50 veces mayor que la del clasificador de BCI2000. Además, al utilizar el módulo de obtención de señales y estimulación propuesto en este trabajo en conjunto con el clasificador, se puede observar un aumento en la exactitud promedio, aunque la principal ganancia en la exactitud promedio se encuentra en el clasificador propuesto, ya que este aumentó dicha métrica en un 26.67%, mientras que el utilizar el módulo de estimulación solo lo hizo en un 1.25%. Sin embargo, esto permite observar que el sistema integrado presenta un mejor desempeño que el ofrecido por BCI2000.

La exactitud promedio de 76.25% obtenida con el sistema propuesto en este megaproyecto equivale a decir que, en promedio entre distintos usuarios, el sistema logrará predecir correctamente 11 de cada 15 letras. Esto podría, en principio, no parecer una exactitud muy elevada. Sin embargo, deben de considerarse varios puntos al momento de apreciar esta métrica: Como primer punto, siendo esta una interfaz no convencional cuya mayor utilidad podría hallarse en permitir la comunicación de personas que no tienen otro medio para escribir o dar comandos a un dispositivo, pasar de la imposibilidad de hacerlo a tener una opción que permita realizarlo, aunque sea de una forma no muy exacta, constituye un gran avance. Como segundo punto, se debe de considerar que la captación e interpretación de señales cerebrales, especialmente de potenciales relacionados con eventos, es un problema sumamente complejo, como lo demuestra la gran cantidad de técnicas (y el costo de ellas, en términos computacionales y del tiempo utilizado) que se deben de aplicar para mejorar la relación de señal a ruido de las señales de EEG. Operar con señales de EEG es más complicado que operar con señales eléctricas provenientes de los músculos, puesto que las señales cerebrales atraviesan el cráneo para poder ser medidas de una forma no invasiva, proceso durante el cual la amplitud de dichas señales se ve considerablemente reducida. Asimismo, las señales cerebrales ocurren dentro de una región que presenta una gran densidad de células nerviosas, por lo que existe una superposición de las señales eléctricas generadas por distintas neuronas y regiones del cerebro, dificultando la detección de componentes específicos de dichas señales. Finalmente, se debe de observar también que el rendimiento de una interfaz cerebro computadora depende considerablemente del usuario. Esto se puede verificar al comparar las exactitudes promedio obtenidas con las cadenas que obtenían sus datos del módulo de obtención de señales de este megaproyecto contra los resultados de las cadenas que utilizan BCI2000: Si bien la exactitud promedio en el primer caso nunca excedió el 77%, en el otro caso se obtuvieron exactitudes promedio de hasta el 85%. Además, al analizar los resultados individuales, se puede observar que hay sujetos para quienes el sistema predice entre 13 y 15 letras correctas de 15, mientras que con otros apenas se logran 7.

Por todo lo anterior, las métricas de desempeño dadas en este capítulo deben de apreciarse más bien como medidas relativas a los sujetos de prueba, métodos y dispositivos utilizados que permiten, dentro de condiciones similares, realizar comparaciones. Y una comparación que presenta la validez y funcionalidad del clasificador propuesto es que este presenta, bajo las mismas condiciones que otro clasificador, un mejor desempeño. Esto concluye que el sistema desarrollado en este trabajo posee un desempeño aceptable y comparable con otros sistemas disponibles.

2. Procesamiento y clasificación en tiempo real. Una vez elegida la cadena de procesamiento a utilizar, la cual fue la cadena C.2, se procedió a realizar una implementación de esta que pudiera ser utilizada en una aplicación de tiempo real. Para lograr esto, inicialmente se realizó el programa de tiempo real utilizando BCI2000 para recuperar los datos del Emotiv EPOC y *FieldTrip* para transferir tales datos a MATLAB. Una vez recibidos los datos, estos eran almacenados en un lista que actuaría como un búfer interno al programa de MATLAB, para no depender del búfer de *FieldTrip*, el cual podría eliminar datos útiles con el paso del tiempo.

Los datos eran agregados a dicho búfer tan pronto como eran reportados por *FieldTrip*: Para ello, dicho programa cuenta con una variable que indica cuál es el número de la última muestra obtenida, mientras que el programa de MATLAB llevaba control del número de la última muestra almacenada. De esta forma, cuando la diferencia de dichas cantidades era mayor a cero, el programa realizado procedía a tomar las muestras de *FieldTrip* y almacenarlas en el búfer interno. Tan pronto como se recuperaban nuestras nuevas, se aplicaba el filtro CAR sobre ellas, el cual se podía calcular únicamente con el conocimiento de las muestras actuales, así como el filtro temporal IIR de orden 4, el cual se podía aplicar conociendo las muestras actuales y las muestras y salidas del filtro de los cuatro tiempos muestreados anteriormente. Por último, se utilizaba el filtrado espacial, el cual podía calcularse conociendo únicamente las muestras actuales. Los resultados obtenidos a la salida de estas tres etapas de procesamiento eran almacenados en otro búfer interno al programa de MATLAB.

Luego de revisar si existían nuevas muestras, el programa procedía a revisar si existían nuevos estímulos. Si dichos estímulos existían y se contaba con el tiempo suficiente posterior al estímulo para analizar la señal (en el caso de la cadena de procesamiento C.2, 800 ms), se procedía a recuperar las señales de todos los canales en dicho tiempo posterior al estímulo y, según la fila o columna que se había iluminado, se sumaban 1/15 de estos datos a un acumulador. El 1/15 era para calcular desde ya el promedio de la respuesta a los estímulos, puesto que se utilizarían 15 estimulaciones por elemento de la matriz. Además, se llevaba un control de cuántas estimulaciones habían sido tomadas entres todas las señales promedio.

Una vez se alcanzaba un número de 180 estimulaciones, se procedía a alimentar los tres clasificadores con las señales promedio calculadas. Luego de sumar las probabilidades de salida dadas por estos clasificadores, se procedía a seleccionar aquella fila y columna con una mayor probabilidad de contener una

onda P300, información con la cual ya se podía predecir una letra. Se esperó un número de 180 estimulaciones puesto que se tenían 12 elementos (6 filas y 6 columnas), los cuales fueron iluminados 15 veces cada uno.

Al finalizar la predicción, se reiniciaban los acumuladores de las señales promedio y del número de estimulaciones y el resto del programa continuaba operando de la forma ya descrita para predecir la siguiente letra.

Una vez verificado que el programa anterior funcionaba utilizando BCI2000 y *FieldTrip*, se incorporó dentro del código del módulo de obtención de señales. Además, se probó el sistema en tiempo real con cinco sujetos de prueba. Estos mismos sujetos habían sido utilizados para la interfaz de P300 en modo de operación offline y en BCI2000. Los resultados se compraran en el siguiente cuadro.

Cuadro 25. Comparación de la exactitude obtenida en diferentes sistemas y modos de operación.

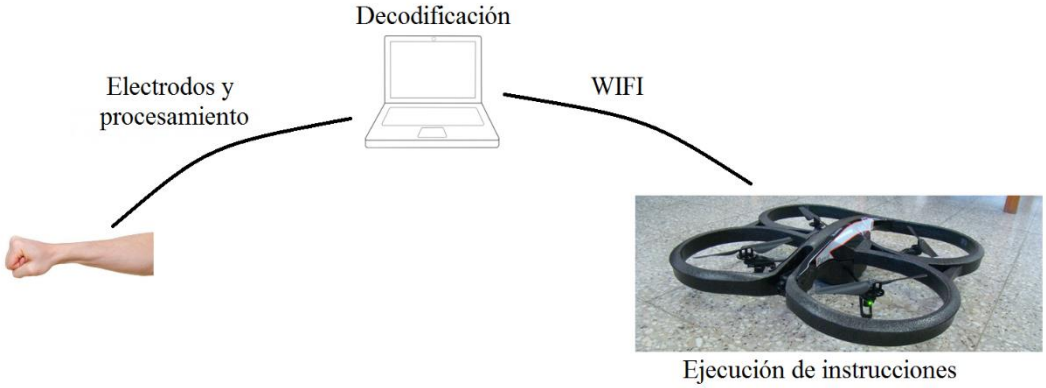
Sujeto	Exactitud Promedio (%)		
	BCI2000 Tiempo Real	Grabación y Clasificador propuesto Offline	Grabación y Clasificador Propuesto Tiempo Real
C.01	60.00	80.00	86.67
C.02	46.67	91.67	86.67
C.03	46.67	60.00	60.00
C.04	46.67	60.00	66.67
C.05	60.00	100.00	95.65
Promedio	52.00	78.33	79.13

Como se muestra en el Cuadro 25, el desempeño de la interfaz desarrollada en este trabajo es mayor al de BCI2000, tanto en modo de operación offline, como online. Estos resultados son consistentes independientemente del sujeto de prueba, lo cual comprueba que el sistema desarrollado en este megaproyecto es funcional.

VI. INTERFAZ BASADA EN SEÑALES ELECTROMIOGRÁFICAS

Como parte del presente megaproyecto, se implementaron electrodos secos para obtener las señales electromiográficas y controlar un drone con tales señales. Para implementar dicho tipo de sistema, se deben realizar las siguientes tareas: Primero, el usuario deberá generar señales electromiográficas específicas (realizar distintos gestos de la mano). Estas señales son filtradas y muestreadas para que una computadora sea capaz de clasificarlas (con un algoritmo de aprendizaje automático) y posteriormente darles un uso (generar movimientos en el drone). Las anteriores etapas son ilustradas en la Figura 98.

Figura 98. Diagrama de bloques para la interfaz basada en señales electromiográficas



A. METODOLOGÍA Y RESULTADOS

1. **Pruebas preliminares con señales EMG.** El circuito para captar señales EMG reales aún no estaba construido. Por lo tanto, para empezar a trabajar con tales señales y realizar pruebas con los modelos de inteligencia artificial seleccionados, se comenzaron a utilizar estos modelos con señales de prueba. Estas son señales EMG tomadas de Beth Israel Deaconess Medical Center (BIDMC), en las cuales se utilizaron electrodos invasivos, tipo aguja, colocados en la pierna de los pacientes. Las muestras utilizadas fueron muestreadas a 4 KHz. Estas fueron tomadas con electrodos invasivos, de tipo aguja, colocados en las piernas de los pacientes. Se trataron tres tipos de pacientes: Saludables, con neuropatía, y con miopatía, ver Figura 99,Figura 100 y Figura 101 [80].

Figura 99. Muestra del músculo tibial anterior, de un hombre de 44 años con enfermedad neuromuscular. Obtenida con Medelec Synergy N2 EMG Monitoring System, utilizando electrodos de aguja [80].

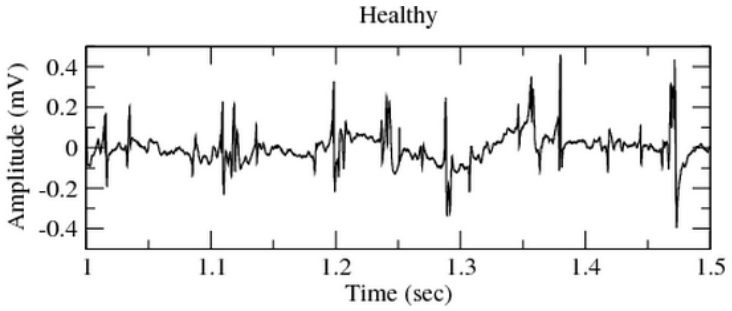


Figura 100. Muestra del músculo tibial anterior, de un hombre de 62 años con dolor lumbar crónico y neuropatía. Obtenida con Medelec Synergy N2 EMG Monitoring System, utilizando electrodos de aguja [80].

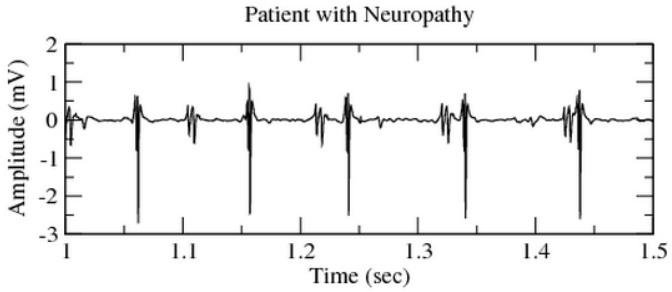
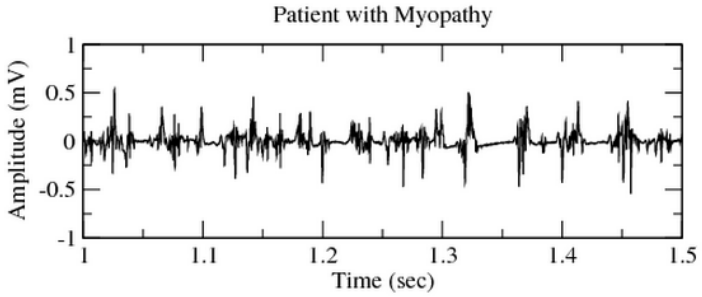


Figura 101. Muestra del músculo tibial anterior, de un hombre de 57 años con miopatía debido a polimiositis. Obtenida con Medelec Synergy N2 EMG Monitoring System, utilizando electrodos de aguja [80].



.

a. Implementación de ANN. Las primeras pruebas se realizaron sin utilizar capas intermedias en la ANN, con una única salida y se realizó en el dominio del tiempo. Para obtener varias muestras se tomaron distintos segmentos de datos de las muestras de cada paciente. Estos fueron tomados de manera aleatoria para variar el desfase entre cada muestra. Se realizó una serie de entrenamientos con distinto número de muestras y cantidad de datos por cada una de ellas. Luego se analizaron nuevas muestras con las redes entrenadas, utilizando un segmento de las muestras principales distinto y se identificó el porcentaje de muestras clasificadas correctamente. Estas pruebas se realizaron tres veces por cada configuración. Se promediaron los datos, para identificar el número de muestras y cantidad de datos que presentan mejores resultados en los entrenamientos, ver Cuadro 26.

Cuadro 26. Porcentaje de aciertos con distinta configuración de datos y muestras.

Muestras	**Cantidad de datos**					**Promedio por muestras**
	100	150	200	300	400	
200	31.69	38.61	35.54	37.31	35.26	35.68
300	36.81	30.90	37.97	36.38	36.94	35.80
400	26.20	34.50	27.67	34.07	36.32	31.75
800	28.04	26.13	25.78	30.38	24.31	26.93
1000	19.54	28.66	27.36	38.65	38.55	30.55
Promedio por datos	28.46	31.76	30.87	35.36	34.27	

Para determinar la cantidad de salidas que presenta mejores resultados se realizó el entrenamiento simultáneo para 1, 2 y 3 salidas, obteniendo un valor decimal y dos binarios respectivamente, siendo el de 2 y 3 salidas los más significativos. Con los entrenamientos simultáneos se agregó un nuevo entrenamiento, aplicando una FFT como entrada, para 3 salidas. Esto se realizó para poder comparar el comportamiento de la red en el dominio del tiempo y de frecuencias. Se realizaron dos barridos, uno de nodos por capa y el otro de capas.

Cuadro 27. Porcentaje de acierto promedio entre 5 y 50 capas, con 100 datos de entradas, evaluado en dominio del tiempo para 2 y 3 salidas para dominio de frecuencias.

Nodos por capa	**3 datos**	**2 datos**	**FFT**
5	10.3	51.2	29.7
10	44.9	42.6	36.8
15	12.5	31.9	39.5
20	59.5	63.0	34.1
25	74.5	60.1	35.6
30	59.8	62.9	36.1
35	56.5	59.2	33.6
40	54.7	59.6	26.0
45	53.5	59.6	36.0
50	59.5	60.2	28.6
55	49.7	61.1	33.4
60	48.4	58.8	30.0
65	47.9	61.5	28.8
70	49.5	63.2	22.7
75	15.6	57.9	24.6
80	50.0	56.1	25.8
85	50.4	58.4	29.2
90	48.8	55.4	24.7
95	46.6	59.4	23.0
100	41.2	55.4	26.8

Cuadro 28. Porcentaje de aciertos para 25 nodos por capa, con 100 datos de entrada, evaluando en dominio de tiempo para 2 y 3 salidas y 3 salidas para dominio de frecuencias.

Capas	3 datos	2 datos	FFT
5	59.5	29.8	45.3
10	69.6	33.3	38.6
15	76.7	36.3	36.3
20	65.5	63.8	38.4
25	56.3	61.4	35.0
30	54.5	60.3	35.5
35	59.8	62.9	34.6
40	58.6	63.1	43.5
45	61.3	62.5	32.7
50	58.2	62.1	41.8
55	53.8	64.2	34.0
60	59.6	62.9	34.8
65	55.2	65.9	37.9
70	51.9	63.8	31.8
75	53.0	65.4	37.9
80	52.8	62.1	35.0
85	62.8	65.2	30.5
90	55.4	39.2	33.6
95	53.5	64.7	33.5
100	52.3	65.9	32.8

Con base en los resultados de los Cuadro 27 y Cuadro 28 se realizaron dos nuevos barridos, en los que se redujeron los rangos a valores cercanos en los que se encontraron los porcentajes de acierto más altos.

Cuadro 29. Parámetros seleccionados para realizar los entrenamientos de la ANN, a 4 KHz.

Capas	**15**
Nodos	25
Entradas	100
Salidas	3
Dominio	tiempo
Porcentaje Máx.	76.70%

b. Implementación de regresión logística. El modelo de inteligencia artificial iba a analizar los datos anteriores. Sin embargo, no se iba a analizar el voltaje o el tiempo en el que ocurrían ciertos picos. Los niveles de voltaje entre una muestra y la otra son muy similares entre sí, por lo que clasificar basándose en niveles de voltajes sería poco eficiente. Se sabe que las señales eléctricas muestreadas varían en: amplitud (poco) y frecuencia. Por lo que se decidió utilizar la Transformada Rápida de Fourier (FFT) para obtener frecuencias de los impulsos anteriores, y se decidió por encontrar el área bajo la curva (potencia de las señales EMG). Esto no se hizo por cada dato, si no que se tomó un segmento de los datos para calcular "n" áreas y "n" frecuencias. En la siguiente tabla se muestran los resultados obtenidos con este regresión logística. La columna de divisiones se refiere a la cantidad de segmentos en las que se partieron los datos (cada arreglo "Healthy", "Myopathy" y "Neuropathy", se dividió en esta cantidad de segmentos). Es decir, cuantas áreas y frecuencias, se calcularon en ese segmento.

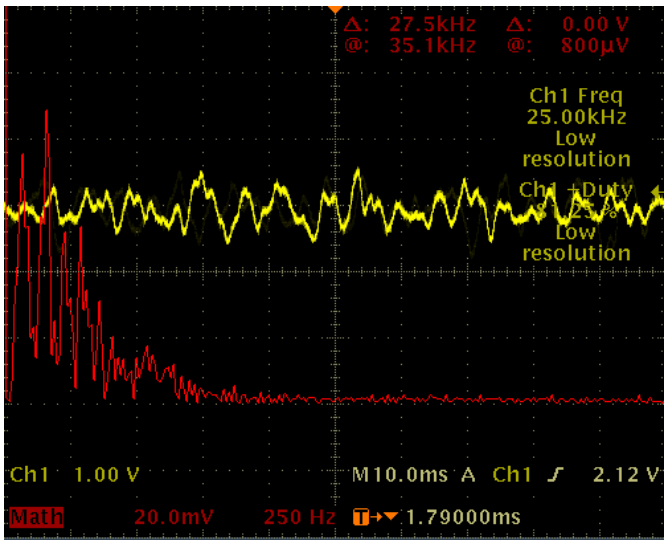
Cuadro 30. Clasificación de pacientes saludables, con neuropatía y con miopatía.

Tamaño de arreglo de pruebas	Tamaño de arreglo de entrenamientos	Divisiones	Exactitud (%)
32000	18950	10	66.67
18950	32000	10	43.33
32000	18950	20	66.67
18950	32000	20	50.00
32000	18950	30	66.67
18950	32000	30	49.33
32000	18950	40	67.50
18950	32000	40	48.57
32000	18950	50	67.33
18950	32000	50	54.00
32000	18950	60	66.67
18950	32000	60	51.67
32000	18950	70	66.67
18950	32000	70	50.00
32000	18950	80	66.67
18950	32000	80	51.67
32000	18950	90	66.67
18950	32000	90	58.89
32000	18950	100	66.67
18950	32000	100	56.33

Los resultados anteriores no fueron aceptables (90% o más de exactitud), por lo que se decidió investigar sobre otro clasificador (SVM) para utilizar este en vez de regresión logística. En ese momento, el circuito para captar señales EMG ya estaba construido, con la limitación de que tenía solo dos electrodos húmedos (necesario utilizar gel en la piel) y el programa para trabajar en tiempo real aún no estaba listo, es decir, no se podía medir y clasificar al mismo tiempo.

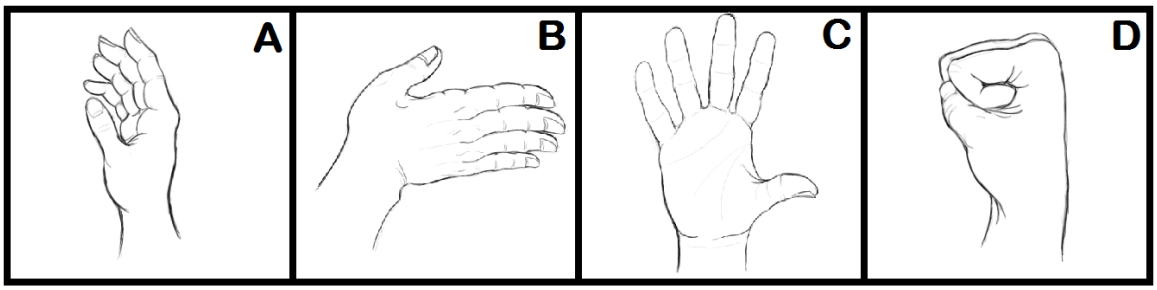
2. Pruebas con señales reales. Se colocaron dos electrodos húmedos en el antebrazo (sobre los músculos extensores y flexores de los dedos, aproximadamente 10 cm del codo) con una separación de 2.5cm. Se buscó esta región ya que tenía una mayor densidad muscular y cubría una mayor cantidad de músculos de la mano. Es importante mencionar que no es necesario llevar a cabo los tres gestos. Es decir, las señales EMG se envían para accionar los músculos (sin importar si estos se lograron accionar o no). Por lo que una persona sin manos (y con nervios funcionando) lograría enviar señales EMG y los clasificadores lograrían interpretar estas señales. Para comprobar esto se sostuvo la mano del usuario mientras el trataba de realizar los gestos. Aun así, las señales EMG variaban y los clasificadores funcionaban de la misma manera. Por lo tanto, esto indica que personas con discapacidades físicas (siempre que las ramas de los nervios se encuentren intacto) podrían crear las señales EMG y estas podrían ser clasificadas. En la siguiente figura se encuentra una de las muestras tomadas con el circuito EMG.

Figura 102. Señales EMG medidas. La gráfica amarilla muestra la señal en función del tiempo y la gráfica roja en función de la frecuencia.



a. Implementación de ANN en circuito EMG. Se tomaron muestras utilizando el circuito EMG para clasificar cuatro gestos en total (ver siguiente Figura 103).

Figura 103. Gestos utilizados para establecer los parámetros de la ANN. Dónde: A, mano relajada; B, músculos extensores de los dedos tensos; C, mano abierta; D, puño con los músculos flexores tensos.



Con el análisis de las señales se determina la ubicación de los electrodos en la que se obtiene una mejor clasificación y se ajustan nuevamente los parámetros de la red neural. Se entrenaron tres ANN, la primera con la señal obtenida de los músculos flexores de los dedos, la segunda con los músculos extensores de los dedos y la tercera con ambas señales como entradas.

Cuadro 31. Porcentaje de aciertos con 76 capas intermedias, 400 entradas y 3 salidas en dominio de frecuencia.

Gesto	Combinado	Flexores	Extensores
A	97.76	84.23	65.04
B	99.14	99.28	84.9
C	79.04	49.9	73.76
D	98.47	47.76	100
Promedio	94.78	71.82	81.62

Para obtener una mejor clasificación se utilizaron dos pares de electrodos, para medir los músculos flexores y extensores simultáneamente, para ello se entrenaron tres ANN, una para los músculos flexores, otra para los extensores y la última involucrando ambas. Al momento de comparar las ANN de los dos conjuntos de músculos por separado, se puede observar que cuando una presenta malas clasificaciones en la otra se obtienen mejores resultados, y viceversa. Esto demostró que las señales se complementan, como se observó en la tercer ANN, que se mantuvo un promedio de aciertos de 94%, ver Cuadro 31.

Cuadro 32. Parámetros a implementar en la ANN, a 10 KHz.

Capas	**15**
Entradas	400
Salidas	3
Dominio	frecuencia
Porcentaje Máx.	94.8%

Al momento de cambiar de las señales de prueba a las señales medidas en el osciloscopio se pudo observar que el tiempo óptimo para ambas clasificaciones se encuentra en el mismo orden de magnitud, 0.025 segundos y 0.040 segundos respectivamente, ver Cuadro 29 y Cuadro 32. A pesar de ello hubo una diferencia significativa al momento de pasar de mejores resultados en función del tiempo, con los datos de prueba, a mejores resultados en función de la frecuencia, con los datos medidos. Esto se debe a que las señales de prueba fueron medidas únicamente sobre un músculo y se realizaba para un único movimiento. Mientras que en las señales medidas se analizó el comportamiento de varios músculos simultáneamente. A partir de ello se puede decir que la clasificación de señales EMG en función del tiempo debe utilizarse para medir los cambios en el comportamiento de un solo músculo, mientras que la clasificación de señales EMG en función de la frecuencia se debe utilizar al momento de analizar el comportamiento de un conjunto de músculos.

b. Implementación de regresión logística y SVM en circuito EMG. Se tomaron muestras con el circuito EMG para tratar de clasificar tres gestos distintos de la mano (ver Figura 104). Posteriormente, se trataría de incrementar la cantidad de gestos posibles de clasificar.

Figura 104. Gestos de manos utilizadas para las primeras pruebas SVM y regresión logística.

Cuadro 33. Exactitudes obtenidas utilizando SVM y regresión logística.

	Regresión logístca	SVM
Divisiones	Exactitud (%)	Exactitud (%)
10	40.00	55.56
20	56.00	57.78
30	42.22	61.48
40	43.33	65.00
50	35.33	66.44
60	36.67	66.85
70	36.67	66.67
80	34.17	66.39
90	34.81	66.54
100	34.33	66.89

Figura 105. Una de las clasificaciones para la posición de “puño” obtenida con regresión logística.

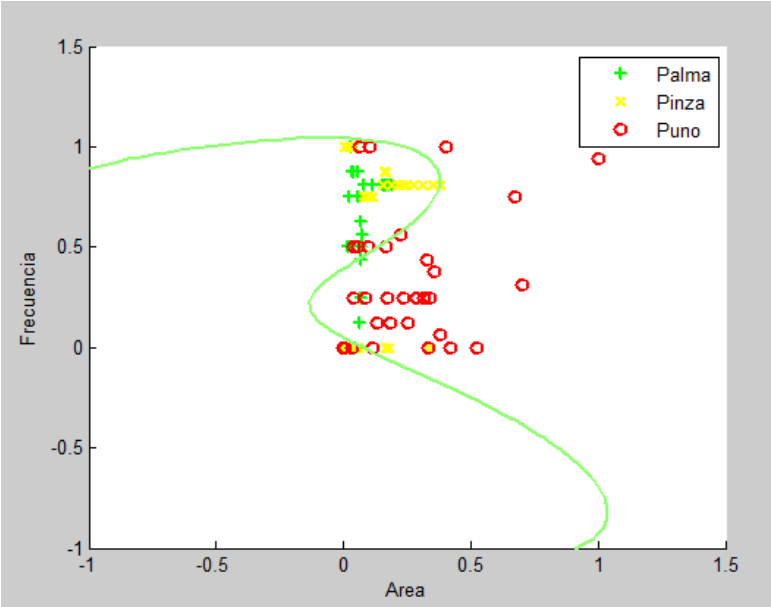
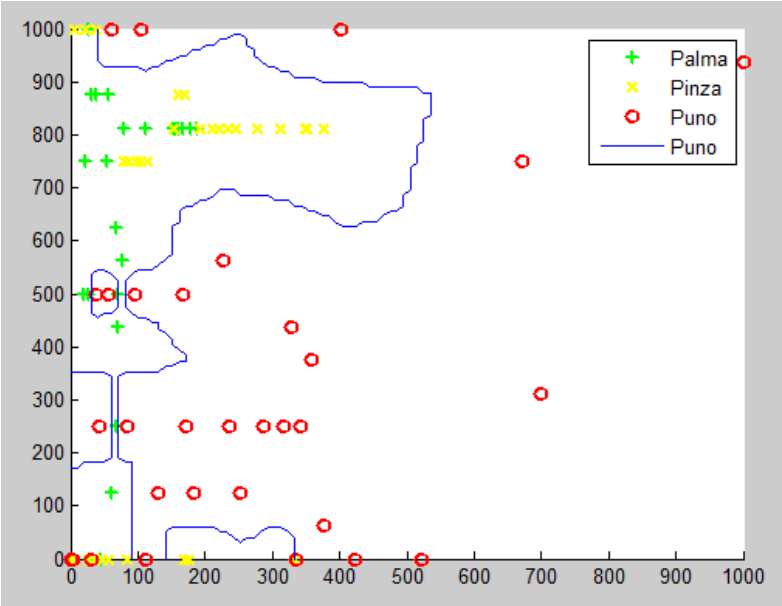


Figura 106. Una de las clasificaciones para la posición de “Puño” obtenida con SVM.



Como se puede apreciar en el Cuadro 33, aunque los dos modelos están clasificando debajo de 90% (lo aceptable), se nota que SVM está desempeñando mejor que regresión logística. Además, si se analizan la Figura 105 y Figura 106, SVM tiene la ventaja de que su curva se puede ajustar a casi cualquier forma y regresión logística no. Debido a esto, se decidió seguir trabajando únicamente con SVM y tratar de mejorar el rendimiento de este. Una manera de mejorar la clasificación es tomar datos intercalados. Es decir, en vez de tomar los datos como $[\text{Dato}_1 - \text{Dato}_{100}, \text{Dato}_{101} - \text{Dato}_{200}, \text{Dato}_{201} - \text{Dato}_{300}, \text{Dato}_{301} - \text{Dato}_{400}, \dots]$, se pueden tomar como $[\text{Dato}_1 - \text{Dato}_{100}, \text{Dato}_{50} - \text{Dato}_{150}, \text{Dato}_{101} - \text{Dato}_{200}, \text{Dato}_{150} - \text{Dato}_{250}, \dots]$. En otras palabras, se toman conjuntos de muestras traslapados. Hacer esto tiene varias ventajas: se tienen más muestras (frecuencias y áreas) con la misma cantidad de datos que antes; y, mientras más muestras existen, más representativos son los resultados y son menos susceptible a datos atípicos). A parte de realizar este proceso, SVM tiene dos constantes C y σ que afectan qué tan sensible es el modelo a datos atípicos. Se variaron estas dos y se compararon los resultados utilizando y sin utilizar el proceso de intercalado. A continuación se muestran los resultados obtenidos con estas modificaciones. En el Cuadro 34. Pruebas realizadas sin traslapar los datos y variando C y σ., la columna que dice “Paso” se refiere al salto de intercalación, es decir, la distancia que hay entre el primer dato de la muestra “n” y el primer dato de la muestra “n+1”. La columna que dice “Tamaño de muestra” se refiere a la cantidad de datos tomados en cuenta para encontrar una muestra (una frecuencia y una área).

Cuadro 34. Pruebas realizadas sin traslapar los datos y variando C y σ.

Datos en total	Divisiones	C	σ	Exactitud (%)
225000	100	0.01	1	66.31
225000	100	0.10	1	62.00
337500	150	0.01	1	77.08
337500	150	0.10	1	77.08
450000	200	0.01	1	54.38
450000	200	0.10	1	75.97
562500	250	0.01	1	76.83
562500	250	0.10	1	63.83
675000	300	0.01	1	76.27
675000	300	0.10	1	63.88
787500	350	0.01	1	76.33
787500	350	0.10	1	76.33
225000	100	0.01	0.1	65.75
225000	100	0.10	0.1	59.94
337500	150	0.01	0.1	55.20
337500	150	0.10	0.1	65.54
450000	200	0.01	0.1	62.66
450000	200	0.10	0.1	67.31
562500	250	0.01	0.1	64.48
562500	250	0.10	0.1	77.60
675000	300	0.01	0.1	76.85
675000	300	0.10	0.1	63.90
787500	350	0.01	0.1	77.27
787500	350	0.10	0.1	64.17

Cuadro 35. Pruebas realizadas traslapando los datos y variando C, σ y el tamaño de la muestra.

Datos en total	Paso	Tamaño de Muestra	C	σ	Exactitud (%)
225000	25	100	0.01	0.1	86.82
225000	25	100	0.10	0.1	79.33
225000	25	100	0.01	1.0	90.96
225000	25	100	0.10	1.0	90.56
337500	25	150	0.01	0.1	85.01
337500	25	150	0.10	0.1	85.23
337500	25	150	0.01	1.0	90.71
337500	25	150	0.10	1.0	90.22
450000	25	200	0.01	0.1	80.07
450000	25	200	0.10	0.1	81.34
450000	25	200	0.01	1.0	91.46
450000	25	200	0.10	1.0	90.26
562500	25	250	0.01	0.1	80.86
562500	25	250	0.10	0.1	88.42
562500	25	250	0.01	1.0	91.94
562500	25	250	0.10	1.0	90.28
675000	25	300	0.01	0.1	87.46
675000	25	300	0.10	0.1	90.27
675000	25	300	0.01	1.0	91.56
675000	25	300	0.10	1.0	95.06
787500	25	350	0.01	0.1	86.14
787500	25	350	0.10	0.1	90.95
787500	25	350	0.01	1.0	95.42
787500	25	350	0.10	1.0	96.71

Comparando los resultados entre los cuadros anteriores, intercalando las muestras mejoró significativamente los resultados. La mejor exactitud obtenida pasó de ser 77.60% a ser 96.71%. Por esto, se decidió seguir con el método de intercalación para las pruebas en tiempo real, es decir, para medir las señales EMG y clasificarlas al mismo tiempo.

3. Comparación de modelos de inteligencia artificial, ANN vs. SVM. Los resultados obtenidos en la sección anterior fueron comparados con otro modelo de inteligencia artificial, SVM. [13] Para ello se utilizaron los mismos datos de entrenamiento, incluyendo los datos preliminares, para identificar cual realiza una clasificación más robusta. Para ello se realizaron nuevas pruebas, con valores obtenidos a tiempo real y clasificaciones simultaneas. Los factores a tomar en cuenta para comparar estos son: tiempo de entrenamiento (cuanto se tarda en entrenar al

modelo); estabilidad en transiciones (cuanto se tarda en clasificar correctamente al cambiar la posición de la mano y mantener esta clasificación constante); y error (que la salida no cambie sin que la entrada cambie). En estas, SVM presentó un mayor tiempo de entrenamiento y pobre estabilidad en las transiciones, tardaba entre 2 y 3 segundos para estabilizarse. A pesar de que este permitió clasificar cinco gestos, se escogió utilizar la ANN debido a que la aplicación requiere un menor tiempo de respuesta.

4. Mejora de estabilidad en transiciones. Una vez obtenidos los parámetros de la ANN, se procedió a realizar pruebas con una mayor cantidad de señales. Estos resultados se analizaron en tiempo real, para evaluar su comportamiento al momento de realizar transiciones entre los distintos gestos. En estas se obtuvo una inestabilidad en la clasificación entre 2 y 5 segundos, es decir que la respuesta del clasificador varió durante este tiempo. Este retraso se debe a que, en la activación trifásica de los músculos, no solo se activan los músculos que van a realizar la acción, sino que también se activan los músculos opuestos a estos, para regular la velocidad de los dedos al moverse. Esto causa que el clasificador lea las señales provocadas por estos músculos y se obtengan malas clasificaciones durante un lapso de tiempo.

Para disminuir el error del sistema se procedió a tomar un mayor número de datos y desfasar el rango de las muestras, a manera de obtener muestras consecutivas, que estuviesen intercaladas entre sí. Implementando esta técnica se logra eliminar las malas clasificaciones, basados en una población de muestras. Esto permitió reducir el tiempo de respuesta entre 1 y 2 segundos, para cuatro gestos distintos. Este problema se resuelve por medio de software, retornando una salida únicamente cuando la clasificación era estable, es decir que el clasificador muestra el mismo resultado. Para ello se compara cada clasificación con las clasificaciones anteriores, esto permite que la muestra de datos sea mayor, sin la necesidad de consumir más tiempo en la toma de datos.

Otro de los factores a considerar fue el tipo de electrodos y su posición sobre el antebrazo, para ello se evaluó tanto las mediciones en dominio del tiempo como en frecuencia. Al comparar las señales sobre la muñeca, ver Figura 108 y Figura 109, con ambos tipos de electrodos se puede observar que su comportamiento en función del tiempo presenta el mismo patrón, mientras que en dominio de la frecuencia hay una diferencia de frecuencias donde se encuentran los máximos. Esto se debe a que las señales no fueron constantes, es decir que las frecuencias variaban respecto al tiempo, pero mostraron las mismas regiones en donde se concentraron estos máximos. Por ello se decidió utilizar los electrodos secos, los cuales requieren de una menor preparación al momento de colocarlos. Por otro lado, se evaluó el comportamiento de las señales en distintas regiones del antebrazo, ver Figura 102. En estas se pudo observar que, en las regiones con mayor masa muscular del antebrazo, se obtenían señales con mayor amplitud, aunque el comportamiento de la señal fue el mismo. A pesar de ello se decidió colocar los electrodos sobre la muñeca, donde los músculos se encuentran más cerca el uno del otro, ver Figura 107.

Figura 107. Brazalete con cuatros electrodos secos con posicionamiento ajustable.

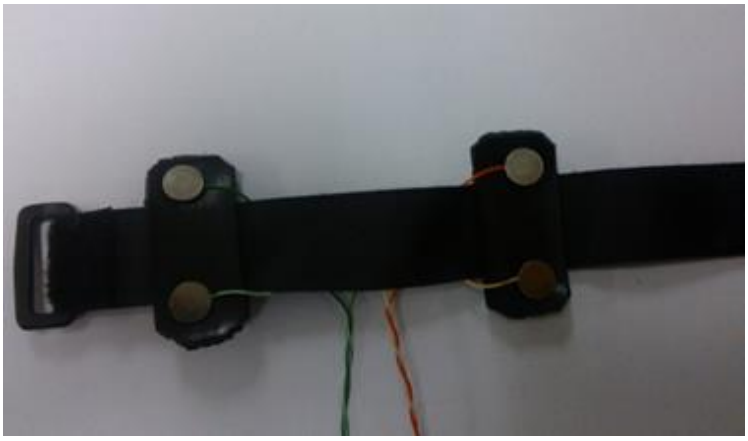


Figura 108. Señal EMG medida con el osciloscopio utilizando electrodos húmedos, sobre los músculos flexores en la muñeca. La gráfica amarilla muestra la señale en función del tiempo y la gráfica roja en función de la frecuencia.

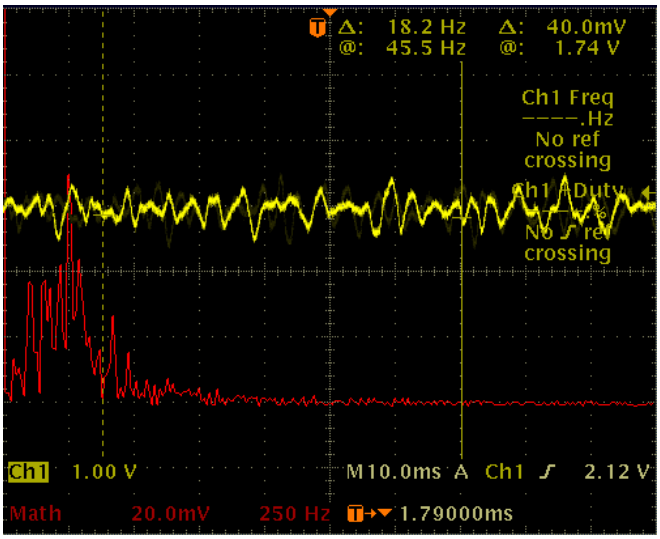
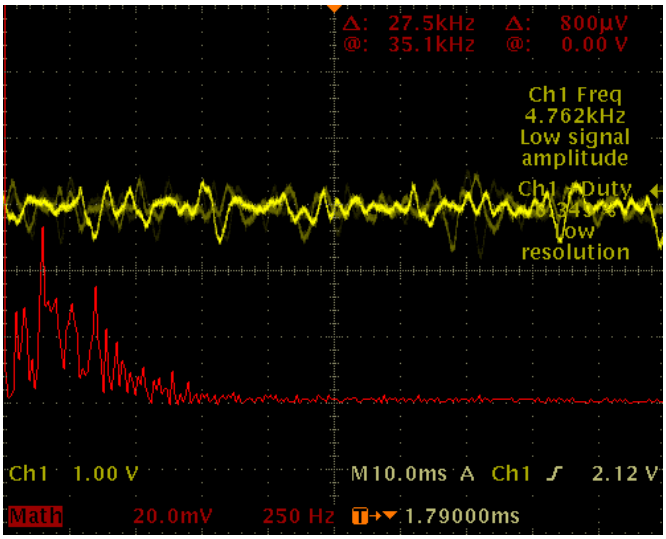


Figura 109. Señal EMG medida con el osciloscopio utilizando electrodos secos, sobre los músculos flexores en la muñeca. La gráfica amarilla muestra la señal en función del tiempo y la gráfica roja en función de la frecuencia.



5. Control básico de AR Drone 2.0. Se conectó a la red inalámbrica creada por el drone. Ya en este, se creó un puerto de comunicación para enviar comandos (puerto 5556). Aún no se estaba tratando de obtener información del drone y no se estaba implementando el control por medio de señales EMG. Primero se realizaron pruebas controlando el drone manualmente. Luego, al estar seguro que los comandos eran los correctos, se implementaría el control por medio de posiciones de la mano. La forma de enviar un comando en MATLAB es la siguiente:

$$\text{sprintf}(\text{AT} * \text{"COMANDO"} = \ \%\text{d}, \%\text{d}, \ldots, \%\text{d}, \backslash\text{r})$$

Donde:

"COMANDO" es una palabra clave que le indica al drone qué tipo de movimiento se hará.

%d es el argumento (en punto flotante). La cantidad de estos varía según el comando.

\r indica que la linea terminó (fin de comando).

Cuadro 36. Sintaxis y función de los comandos implementados para controlar el drone.

AT*"COMANDO"	Secuencias	Función
AT*FTRIM	= %d, %d,\r	Establece la referencia horizontal del drone.
AT*REF	= %d, %d, %d, %d, %d, %d,\r	Despegue, aterrizaje, parada de emergencia.
AT*PCMD	= %d, %d, %d, %d, %d, %d,\r	Sobre volando, Giro, traslación horizontal y vertical.
AT*COMWDG	= %d, %d,\r	Reinicia el watchdog.

Es importante mencionar que para establecer la referencia horizontal, es necesario que el drone se encuentre en el suelo. También es importante mencionar que el primer argumento que recibe el drone es un número de secuencia. Con esto se refiere al número de instrucción que se debería de estar ejecutando. Este número siempre comienza en cero (con el comando de FTRIM) y aumenta con cada comando enviado. Si se llega a superar las 255 instrucciones, es necesario restablecer el watchdog para reiniciar el número de secuencia y así poder enviar más comandos. Si se envía un comando arriba de 255 o se envía un comando debajo del número de secuencia actual, se ignora esa instrucción. También es importante mencionar que, si se pasan más de 2 segundos sin enviar un comando, el drone finaliza la comunicación con el usuario. Para corregir esto, es necesario restablecer el watchdog. Entre cada instrucción enviada, debe de existir una pausa de al menos 0.3 segundos. Esto se debe a dos razones: darle tiempo al drone de ejecutar el comando (así no se interrumpe a la mitad de un comando anterior); y, debido a que se utiliza el protocolo UDP para comunicarse con el drone, no se puede asegurar que el primer paquete de datos enviado será el primero que recibirá el drone, o que el drone guardará el segundo comando recibido en su buffer en vez de tratar de ejecutarla cuando sigue efectuando una instrucción anterior, o que siquiera ejecutará la instrucción. Si se envía un comando erróneo, el drone no envía una alerta de lo anterior, solo ignora la instrucción. Luego de poder enviar comandos sin tener problemas de comunicación, se notó que la secuencia prestablecida para aterrizar al drone, baja al drone bruscamente. Por esto, se decidió tratar de leer de los sensores ultrasónicos y

enviar el comando de aterrizaje cuando el drone estuviera en una posición cercana al suelo (de esta manera se evita que golpee el suelo con mucha velocidad). Como el drone tiene cuatro puertos con funciones específicas, se tuvo que habilitar el puerto por el cual se envían datos al usuario (puerto 5554). Se debe mandar un comando al drone indicando que el usuario quiere los datos: $\mathrm{fread}(\mathrm{Puerto}, 292, 'uint8')$. El drone no envía un dato en específico, si no que envía todos los datos que tiene a su disponibilidad. Uno recibe un arreglo de 500 caracteres. Dentro de estos, cuatro indican la altura a la que se encuentra el drone (posiciones 41, 42, 43, 44). [37] El conjunto de estos cuatro indica la altura, pero el drone lo separa de esta manera por el tamaño de este dato. Debido a esta separación, los valores en estas celdas no indican la altura directamente, sino que estos se deben de convertir de un número decimal a uno binario para luego concatenarlos y tener un dato de 32 bits. Este luego se convierte a un número decimal y se tiene la altura en milímetros.

La implementación del control con señales EMG consistía en ejecutar el siguiente ciclo: muestrear señales EMG, clasificarlas (ver qué instrucción se debía ejecutar), y enviar la instrucción al drone. Este ciclo se tardaba más de 0.3 segundos, por lo que el drone no se movía de forma continua. Si la pausa entre las instrucciones enviadas es mayor de 0.3 segundos, el drone ejecuta un comando predeterminado. En este caso, se queda en el estado de sobre vuelo (flotando en el aire) hasta recibir una nueva instrucción. Ejemplo: Si la posición de la mano era tal que se le estaba indicando al drone que girara constantemente, él giraba 15º y ejecutaba la instrucción de sobre vuelo mientras esperaba la siguiente instrucción (seguir girando). Era hasta ese momento en el que volvía a girar otros 15º. Para evitar esto, se hicieron tres modificaciones para que el movimiento fuera más fluido:

- Debido a que ya existe una pausa en el programa (en lo que ejecuta las demás líneas de instrucción), se redujeron las pausas de 0.3 segundos que se colocaron después de cada instrucción. De esta manera se compensaban estos 0.3 segundos con lo que el programa tardaba en enviar un nuevo comando.
- Reducir la velocidad con la que el drone ejecuta los movimientos. Ejemplo: La velocidad con la que el drone se eleva o desciende estaba configurado en 100% de potencia. Este se redujo a 50% para que los cambios entre el estado de “sobre vuelo” y el estado “subiendo” o “descendiendo” no fueran tan bruscos como antes.
- Enviar varias veces la misma instrucción. Es decir, modificar el ciclo a ser: muestrear señales EMG, enviar la instrucción anterior del drone, clasificar las señales EMG, y enviar la instrucción actual del drone. El hecho de que se está enviando la instrucción anterior no afecta de manera significativa. Esto terminaría enviando el comando una vez más a la deseada. Además, se decidieron colocar los comandos en este orden porque en donde más se tarda el programa es en muestrear las señales y en operar matrices (por lo que también se envió la instrucción justo después de esta operación).

Figura 110. Control con señales EMG en funcionamiento.

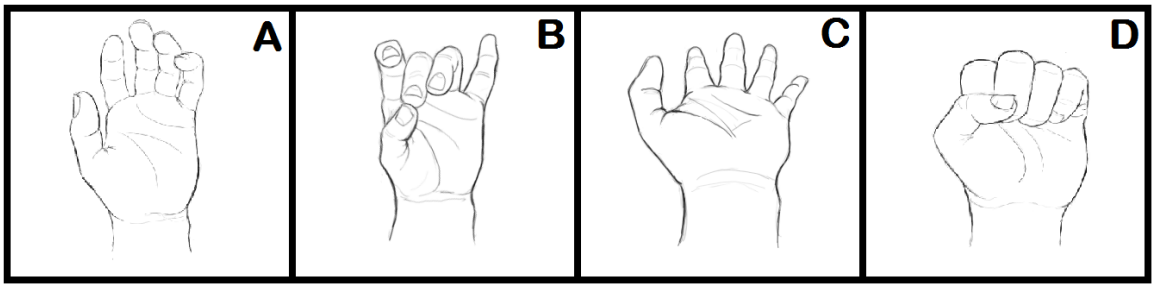
6. Especificación de gestos para controlar AR Drone 2.0. Para determinar la funcionalidad de la interfaz basada en señales EMG, se desarrolló una aplicación de vuelo para el AR. Drone 2.0. En la cual se controla tanto su posición en tres dimensiones, como su orientación. La aplicación se diseñó para obtener datos del clasificador a tiempo real, por lo que se tomó en consideración el tiempo de respuesta del sistema y la estabilidad de los datos obtenidos del clasificador. También se tomó en consideración la cantidad de comandos mínima que se necesitan para controlar el drone y realizar las rotaciones y traslaciones adecuadas.

Se definieron los comandos necesarios para controlar el A.R. Drone 2.0 en pleno vuelo, para esto se definió que la mano izquierda daría el control de altitud, ver Figura 111, mientras que la mano derecha controlaría el movimiento sobre un plano horizontal, ver Figura 112. Con la condición de que el drone solo se pusiese mover hacia adelante, donde tiene ubicada la cámara. Al momento de entrenar con distintas señas, se buscó implementar las cuatro más intuitivas para cada mano, las necesarias para obtener un control total del A.R. Drone 2.0. En estas se buscó que distintos músculos estuviesen involucrados en cada gesto y así obtener una mejor clasificación. Los comandos se definieron como:

Mano izquierda:

- Neutro: mantiene una altura constante, independiente de la posición horizontal del drone.
- Subir: aumenta la altitud del drone gradualmente para tener un mejor control por parte del usuario.
- Bajar: disminuye la altitud del drone gradualmente para tener un mejor control por parte del usuario.
- Aterrizar: suspende las pruebas y desciende gradualmente hasta una altura predefinida, antes de aterrizar.

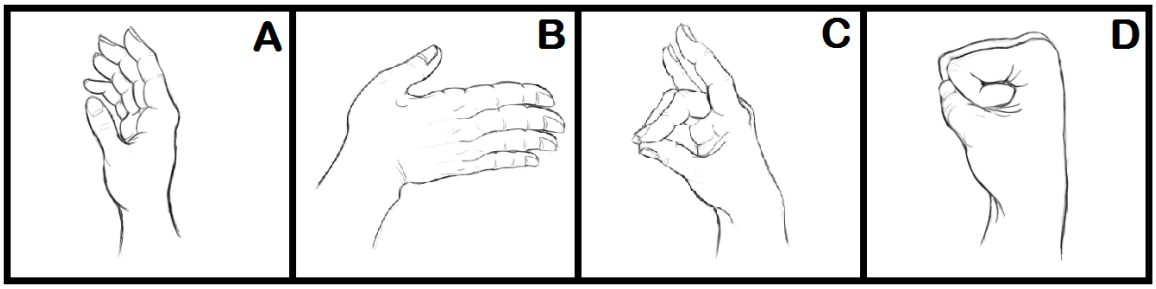
Figura 111. Dibujos de las vistas superiores de las posiciones a utilizar en el entrenamiento con la mano izquierda. Posición de la palma orientada hacia arriba: A (Neutro), posición con la mano relajada; B (Subir), posición de pinza, con los músculos flexores de los dedos medio y anular tensionados; C (Bajar), músculos extensores de los dedos tensionados; D (Aterrizar), mano empuñada, con los músculos flexores de los dedos tensionados, aplicando mayor fuerza en los dedos anular y meñique.



Mano derecha:

- Neutro: mantiene una posición constante sobre el plano horizontal.
- Giro derecho: mantiene su posición sobre el plano y cambia su orientación gradualmente, con una rotación negativa en Yaw.
- Giro izquierdo: mantiene su posición sobre el plano y cambia su orientación gradualmente, con una rotación positiva en Yaw.
- Avanzar: se traslada sobre su eje local X, hacia el frente, con la misma orientación de su cámara.

Figura 112. Dibujos de las vistas superiores de las posiciones a utilizar en el entrenamiento con la mano derecha. Posición de la palma orientada hacia el torso: A (Neutro), posición con la mano relajada; B (Giro hacia la derecha), músculos extensores de los dedos tensionados; C (Giro hacia la izquierda), posición de pinza, con los músculos flexores de los dedos medio y anular tensionados; D (Avanzar), mano empuñada, con los músculos flexores de los dedos tensionados, aplicando mayor fuerza en los dedos anular y meñique.



7. Implementación de interfaz basada en señales EMG. Para determinar si la interfaz es lo suficientemente intuitiva y amigable con el usuario se recurrió a hacer pruebas con estudiantes ajenos al proyecto. Con esto se buscó que no hayan estado presentes en pruebas previas de la interfaz, sino que aprendieran a utilizarla en el momento. Las pruebas consistieron en tres diferentes etapas: entrenamiento de las señales EMG, retroalimentación visual de los comandos y pruebas de vuelo.

- Primera fase: entrenamiento de las señales EMG, se explica a los sujetos de pruebas como colocarse los brazaletes y la orientación para colocar las manos. Se corre el programa de entrenamientos, donde se le indica paso a paso que mano y comando estaba entrenando. Con esto se pretende familiarizar al usuario con la funcionalidad de cada mano y gesto, mientras se recolectan los datos para entrenar la ANN de cada mano.
- Segunda fase: se realiza una prueba a tiempo real, donde se le muestra al sujeto de pruebas, que comando es el que tiene seleccionado. Debido a que los entrenamientos requieren se cierta fuerza en músculos específicos, se le pide al sujeto que regule la fuerza de sus diferentes dedos hasta ver los comandos que él está ingresando en la pantalla. Esto se le pide para cada comando y mano. Una vez realizada esta prueba se le pide que se recuerde de la fuerza aplicada en cada comando y que realice nuevamente el entrenamiento. Se repite la prueba hasta que el sujeto esté conforme con sus resultados, entre 2 y tres veces.
- Tercera fase: se procede a controlar el A.R. drone 2.0 en la cual se le pide al sujeto realizar una serie de comandos, para evaluar si el entrenamiento fue el realizado correctamente y se da un tiempo para poder familiarizarse con los comandos en pleno vuelo.

Figura 113. Sujeto de prueba con posición de manos en neutro.

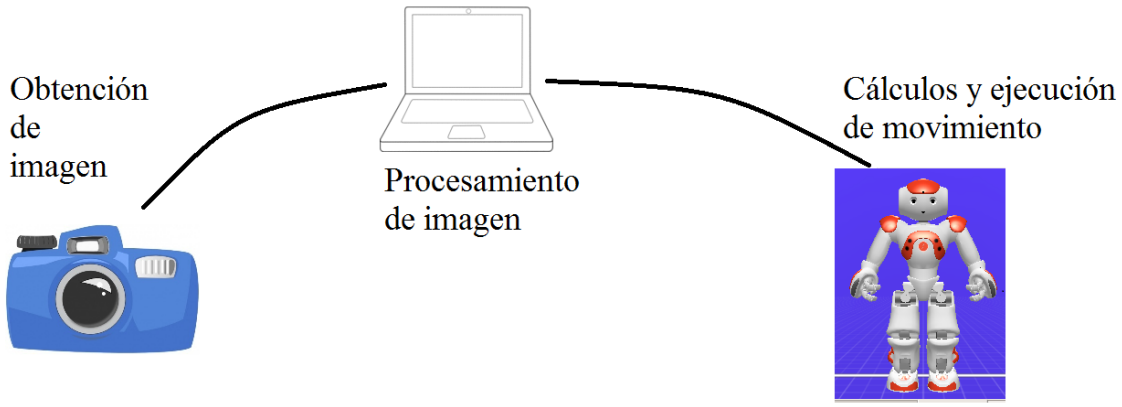
Al momento de realizar las pruebas de la interfaz, se decidió probar la interfaz con personas que nunca estuviesen presente en pruebas previas; esto se decidió para que la interfaz fuese completamente nueva para los usuarios. Luego del primer entrenamiento se observó que los sujetos de pruebas no controlaban bien los comandos, esto se debe a que no mantenían una tensión constante en los mismos músculos al momento de

realizar las pruebas. Lo cual se pudo mejorar dándoles una retroalimentación visual de las salidas del clasificador. La retroalimentación causó una mejora significativa en los resultados debido a que los usuarios podían ajusta la fuerza de sus músculos para obtener las salidas deseada. Aunque los resultados mejoraron, se les pidió realizar el entrenamiento y las pruebas con retroalimentación nuevamente, de esta forma se puede crear una memoria muscular en los sujetos, la cual es más fácil de identificar. El tiempo en que los usuarios aprendieron a utilizar la interfaz se mantuvo cerca de 20 minutos, aunque es recomendable aumentar la cantidad de entrenamientos para obtener mejores resultados.

VII. INTERFAZ BASADA EN MOVIMIENTOS GRUESOS (VISIÓN POR COMPUTADORA)

Como parte del presente megaproyecto, se realizó la implementación de un sistema de visión por computadora para obtener las coordenadas cartesianas de las manos del usuario y posteriormente realizar los cálculos respectivos para que un robot humanoide (NAO) replique los movimientos del usuario. Para implementar dicho tipo de sistema, se deben realizar las siguientes tareas: Primero, tomar una foto del usuario y procesar esta imagen (escalarla e identificar las coordenadas de las manos). Luego de obtener estas coordenadas, se ingresan estas en un método numérico que encuentra los ángulos necesarios que deberá tener cada motor para que las extremidades de NAO se encuentren en la misma posición que el usuario. Estos ángulos se envían a NAO para que él actualice la posición de sus manos. Las tareas anteriores son diagramadas en la Figura 114.

Figura 114. Diagrama de bloques para interfaz basada en movimiento en gruesos



A. METODOLOGÍA Y RESULTADOS

1. Desarrollo de interfaz basada en visión por computadora

a. Parámetros para filtrado de colores. Para identificar las articulaciones del cuerpo humano se recurrió a utilizar esferas plásticas como marcadores, para estas se buscó utilizar colores con un alto índice de reflexión y colores poco utilizados en los entornos. Para identificar los parámetros del filtrado primero se recurrió a comparar una imagen en donde aparecieran las esferas, con filtros para imágenes RGB, HSV, YCbCr y L*a*b*, utilizando el software Color Thresholder de Matlab, ver Figura 115, donde se obtuvieron mejores en HSV y L*a*b*. De estos se escogió HSV y se realizó un filtrado preliminar, ver Figura 116.

Figura 115. Selección de parámetros HSV por medio de Color Tresholder de Matlab. El valor de H está definido por un círculo de circunferencia igual a uno.

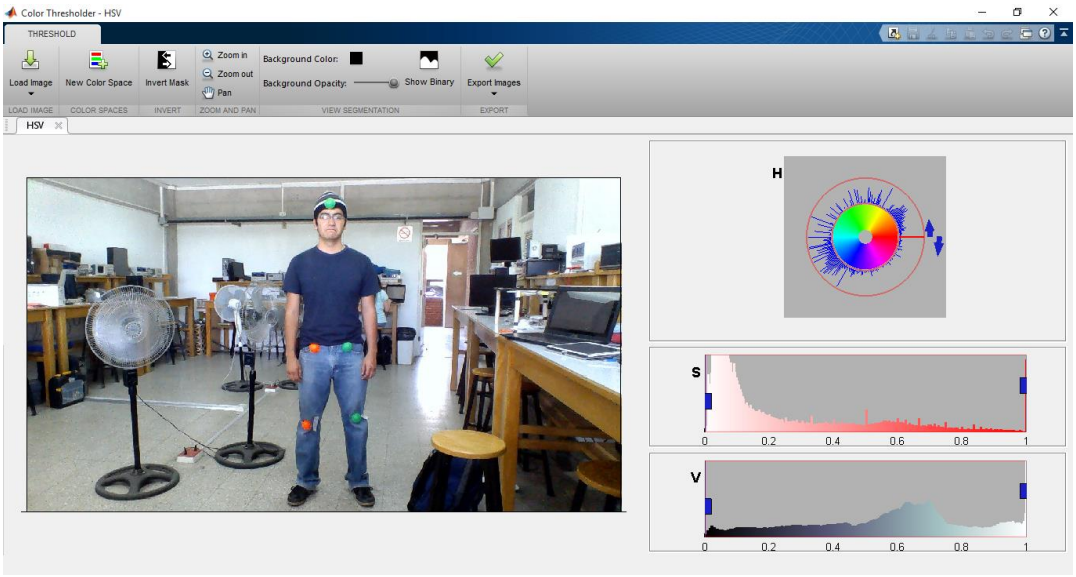


Figura 116. Aplicación del filtrado HSV para identificar las esferas. La imagen de la izquierda muestra las esferas anaranjadas y la de la derecha las esferas verdes.

Debido a que la iluminación del ambiente varía respecto a las diferentes horas del día, se realizó el filtrado para imágenes tomadas en la mañana, tarde y noche. Con esto se definió un rango para cada parámetro H, S y V, para ambas esferas.

Cuadro 37. Rango para los parámetros HSV del filtro de color.

	Anaranjado		**Verde**	
	Min	Max	Min	Max
H	0.00	0.20	0.25	0.55
S	0.40	1.00	0.20	1.00
V	0.70	1.00	0.40	1.00

Al momento de realizar las pruebas de color, fue muy importante tomar en cuenta la iluminación y las sombras provocadas por la misma iluminación. Esto se debe a que el rango de colores varía más en los lugares donde hay reflejo y sombra propia. Por lo que se tuvo que decidir entre reducir el rango de colores y reducir la forma circular de las esferas o aumentar el rango e introducir ruido en las imágenes. Al principio se optó por aumentar el rango, lo cual presento mejores resultados. A pesar de ello no se logró diferenciar la geometría de las esferas en lugares como las rodillas, al momento en que los brazos aumentaban sombra sobre estas. Esto se pudo solucionar agregando iluminación sobre estas, haciendo el sistema más robusto. La iluminación adicional es necesaria dependiendo del entorno en el que se utiliza la interfaz.

b. Identificación de articulaciones en el cuerpo humano. Para identificar cada articulación se compararon dos métodos de reconocimiento de imágenes. El primero está basado en el tamaño de una figura y el segundo está basado en la geometría de una figura. Para el primero se determinó un rango, cantidad de pixeles del color seleccionado, con el cual se buscó eliminar las manchas del fondo con colores similares. En el segundo se estableció un rango de radios, para todas las figuras circulares con colores similares. Este método no requiere de círculos completos, ni que el contorno sea continuo, a diferencia del primer método.

c. Captura de posición de las articulaciones. El programa de reconocimiento de patrones circulares identifica las geometrías y las ordena en una matriz de coordenadas. Este los ordena conforme a su orden de izquierda a derecha y luego de arriba para abajo, ya que se graba a la persona de frente, al lado izquierdo de cada imagen se observa el lado derecho de la persona, ver Figura 118. Para identificar cada punto, se reorganizan los datos de la imagen con filtro verde de arriba para abajo y así obtener las coordenadas de las rodillas. Una vez identificadas las rodillas, se vuelven a ordenar los datos de izquierda a derecha, obteniendo codo derecho, hombro derecho, cabeza, hombro izquierdo y codo izquierdo. La posición de las manos se obtiene de la imagen con filtro anaranjado. Se utilizó un color distinto para las manos, ya que estas se pueden cruzar con las otras articulaciones y causar una desorientación en el programa.

Figura 117. Captura de imagen con aplicación de filtros HSV y reconocimiento de patrones circulares.

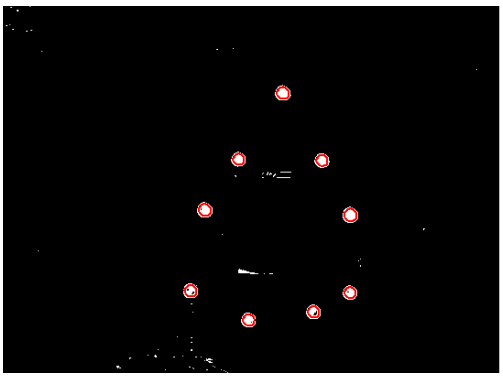
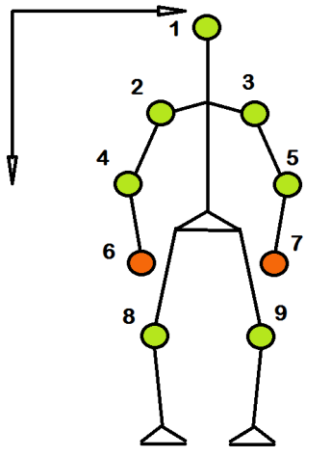


Figura 118. Esqueleto visto de frente, formado por las posiciones de las esferas, el número indica el orden de las articulaciones.



Las posiciones Y y Z son obtenidas por medio de la cámara, pero es necesario determinar la profundidad a la que se encuentra cada articulación, para ello se define que los hombros se encuentran en x=0 y se determina la profundidad a partir de ellos. Esto se realiza por medio de proyecciones, ver Figura 119, por lo que es necesario obtener una imagen de referencia o calibración, ver Figura 120. En esta se requiere que el usuario tenga totalmente extendidas sus extremidades, sobre un mismo plano. Luego se miden las longitudes de las articulaciones, L, y se utilizan de referencia para determinar las profundidades, X, de cada articulación.

Una de las alternativas que se tomó en consideración para determinar la profundidad en la que se encuentran las articulaciones fue el diámetro de las esferas percibido por la cámara. El cual incrementa a medida en que la esfera se acerca a la cámara. El inconveniente con esto fue que el diámetro varía según la iluminación y que la resolución de la cámara no permite distinguir entre profundidades similares, por lo que funciona mejor mientras más cerca este el usuario de la cámara. Caso contrario ocurre al utilizar los largos de las extremidades, las cuales funcionan mejor a mayor distancia. Esto se debe a que la cámara tiene una vista cónica y distorsiona las dimensiones a distintas profundidades. Este efecto se reduce a medida que se aumenta la distancia hacia la cámara.

Figura 119. Proyección de las articulaciones sobre el plano Y-Z. Donde L es el largo de las extremidades, K es la componente proyectada sobre el plano y X es el desplazamiento hacia el frente de la articulación.

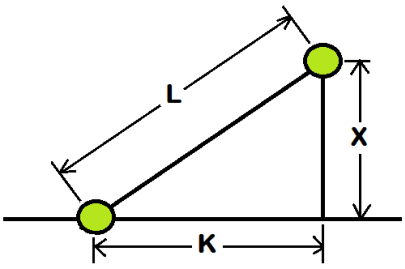


Figura 120. Posición de las extremidades para realizar la calibración.

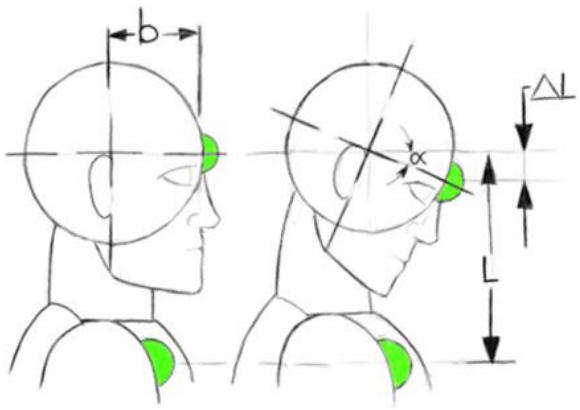
d. Rotación de la cabeza. Para definir la rotación de la cabeza, se calcula el promedio entre los hombros, punto de referencia. Sobre este punto se determina la rotación sobre el eje Z de la cabeza. Para ello se utilizó la diferencia entre la posición en Y de la cabeza y la posición en Y del punto de referencia como X y L como la distancia entre la esfera al centro de la cabeza, ver Figura 119. Para determinar la rotación sobre el eje Y se define la distancia L al momento de calibrar, ver Figura 121. Luego se compara con la distancia L' a tiempo real. Con estos valores se determina α.

$$\Delta L = L' - L$$

$$\Delta L = b * \sin(\alpha)$$

$$\alpha = \sin^{-1}\left(\frac{\Delta L}{b}\right)$$

Figura 121. Análisis de la geometría de la cabeza, para determinar su rotación vertical. Donde b es la distancia desde el centro de la cabeza hasta el centro de la esfera; L, es la distancia vertical entre los hombros y la cabeza, cuando se esta viendo hacia el frente; y ΔL es el desplazamiento vertical de la cabeza.



2. Desarrollo de control de NAO

a. Cinemática inversa de NAO. Se comenzó con modelar a NAO como una cadena (conjunto de eslabones y articulaciones). A pesar de que la interfaz obtiene datos de todo el cuerpo, incluyendo las piernas, se puede perder el equilibrio de NAO si se mueven las piernas. Debido a esto, únicamente se modelaron los brazos y la cabeza de NAO.

Se utilizaron los parámetros Denavit-Hartenberg para simplificar este análisis. A continuación, se muestran cuadros donde se especifican qué valor se utilizaron en cada uno de estos parámetros, junto con los límites máximos de NAO. La columna de "Eslabón" indica el nombre del eslabón. La variable σ indica si la articulación es de tipo cilíndrico (0) o prismático (1). En la columna de "Ángulo máximo", aparece "N/A" porque son eslabones intermedios (eslabones ficticios que se utilizaron para interconectar dos eslabones reales de NAO). Las filas que están en gris indican que estas ya no se tomaron en cuenta para el análisis debido a que son eslabones linealmente dependientes (uno depende del anterior), por lo que solo se incluyó el eslabón anterior y se agregó el desfase adecuado.

Cuadro 38. Parámetros Denavit-Hartenberg para el brazo izquierdo de NAO.

Eslabón	θ (grados)	d (mm)	a (mm)	α (grados)	σ	Ángulo máximo
IntermedioLA1	90	100.00	98.00	90	0	N/A
LShoulderRoll	ThetaLSR – 90	0.00	0.00	0	0	-18 a 76
IntermedioLA2	0	0.00	0.00	-90	0	N/A
LShoulderPitch	ThetaLSP – 90	0.00	0.00	0	0	-119.5 a 119.5
IntermedioLA3	0	15.00	105.00	90	0	N/A
LElbowRoll	ThetaLER	0.00	0.00	90	0	-88.5 a -2
IntermedioLA4	90	0.00	0.00	-90	0	N/A
LElbowYaw	ThetaLEY	113.70	0.00	0	0	-119.5 a 119.5
LWristYaw	ThetaLWY	0.00	0.00	0	0	-104.5 a 104.5
LHand	ThetaLH	0.00	12.31	0	0	N/A

Cuadro 39. Parámetros Denavit-Hartenberg para el brazo derecho de NAO.

Eslabón	θ (grados)	d (mm)	a (mm)	α (grados)	σ	Ángulo máximo
IntermedioRA1	-90	100.00	98.00	-90	0	N/A
RShoulderRoll	90 + ThetaRSR	0.00	0.00	0	0	-76 a 18
IntermedioRA2	0	0.00	0.00	-90	0	N/A
RShoulderPitch	ThetaRSP – 90	0.00	0.00	0	0	-119.5 a 119.5
IntermedioRA3	0	-15.00	105.00	90	0	N/A
RElbowRoll	ThetaRER	0.00	0.00	90	0	2 a 88.5
IntermedioRA4	90	0.00	0.00	-90	0	N/A
RElbowYaw	ThetaREY	-113.70	0.00	0	0	-119.5 a 119.5
RWristYaw	ThetaRWY	0.00	0.00	0	0	-104.5 a 104.5
RHand	ThetaRH	0.00	12.31	0	0	N/A

Figura 122. Modelo de brazo izquierdo en MATLAB.

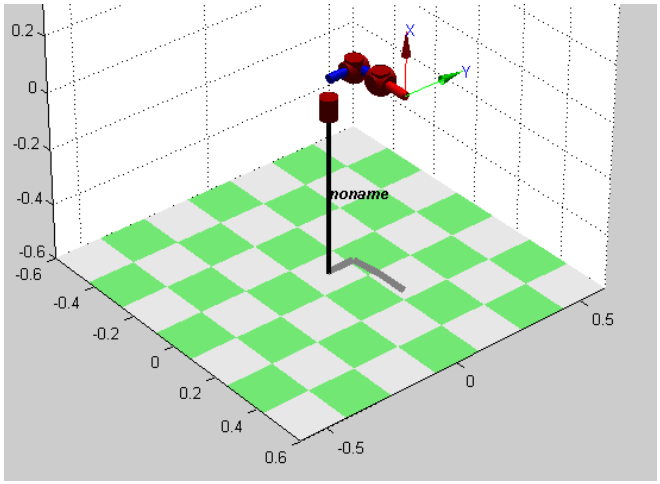
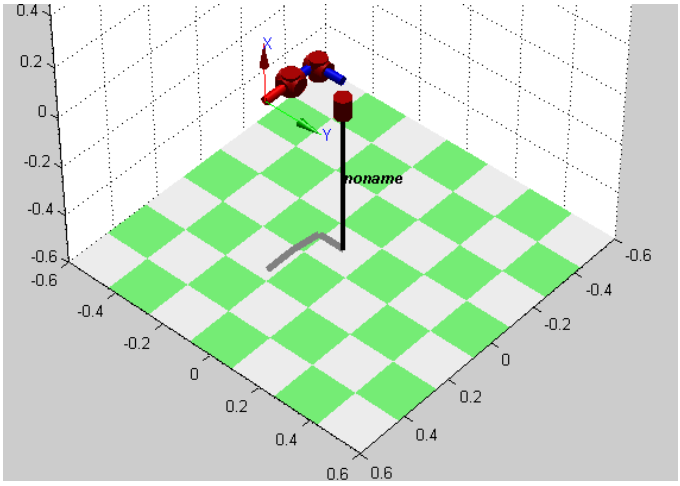


Figura 123. Modelo de brazo derecho en MATLAB.



Para la cabeza no fue necesario establecer estos parámetros debido a que el sistema de visión por computadora devolvía el ángulo de inclinación de la cabeza (por lo que solo se mandaba este dato a NAO). Para poder encontrar los ángulos necesarios para que los brazos estuvieran en posiciones específicas, se utilizó un método numérico. Se sabe que las constantes K, Q y B tienen significados físicos. Sin embargo, si se analizan las ecuaciones, estos únicamente varían un factor proporcional de $\dot{q}\langle k\rangle$. Para facilitar la calibración de NAO y establecer el parámetro K, se configuraron los parámetros Q y B iguales a 1. De esta manera se tendría un único factor que calibrar.

b. Movimiento manual de las extremidades de NAO. Para comprobar que el modelado de NAO era el correcto y que se podría llevar a cabo algún movimiento deseado, se creó el programa PasoAPaso.m. Este une la cinemática inversa con el simulador de NAO (Choregraphe). Básicamente, se ingresan las coordenadas (x,y,z) deseadas para ejecutar la cinemática inversa en MATLAB y simular el movimiento de NAO en Choregraphe.

En la simulación, se notó que NAO se tardaba demasiado en llegar de su posición inicial a la posición deseada. La distancia entre una posición y la siguiente posición era muy pequeña (el paso entre una posición y la siguiente posición era muy grande). Este problema no se debe a la velocidad máxima de NAO debido a que los pasos provienen del método numérico. Por lo tanto, el problema se debe a que el método numérico se acercaba lentamente a la posición deseada. Para acelerar este proceso (aumentar el paso), se aumentó el factor proporcional K. Hacer esto ayudó con la velocidad, pero resultó generando un sobrepaso. Es decir, las extremidades se pasaban de la posición deseada (el paso era muy grande). Por lo tanto, se disminuyó este factor proporcional y se aumentaron las iteraciones del método numérico antes de enviarle las posiciones a NAO. Básicamente, se calculó la cinemática inversa varias veces, lo cual daría como resultado ángulos más cercanos a los deseados (realizar muchos pasos pequeños antes de enviar datos a NAO). Esto resultó en aumentar tiempo de computación (cuánto se tarda la computadora en realizar dichos cálculos) y disminuir el tiempo de simulación (cuánto se tarda NAO en moverse de un punto a otro).

A pesar de estas modificaciones, las iteraciones necesarias para llegar a la posición deseada eran demasiadas (aproximadamente 1000 iteraciones). Este problema se debe a que la posición inicial está lejos de la posición deseada. Para solucionar este problema, se movió la posición inicial más cerca de la posición deseada. En el sistema basado en visión por computadora, el usuario comienza con los brazos apuntando hacia los pies. Esto en NAO resulta en desfase de 90º en los hombros en vez de 0º (ver Figura 124 y Figura 125).

Figura 124. NAO con los hombros desfasados a 0º.

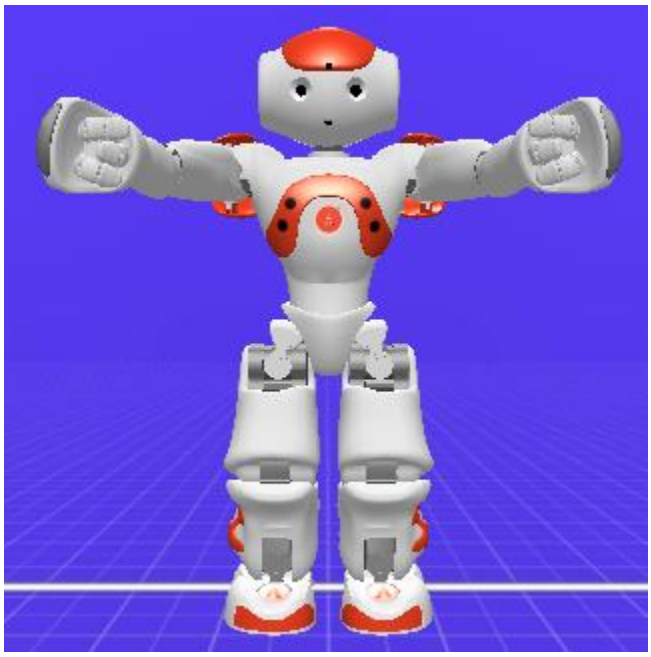
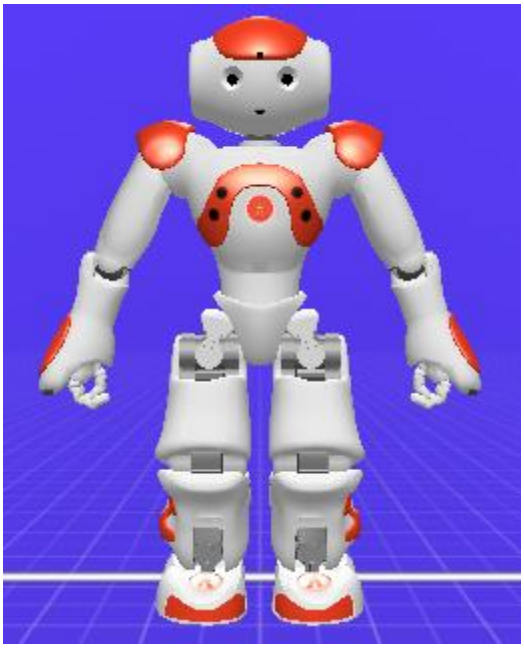


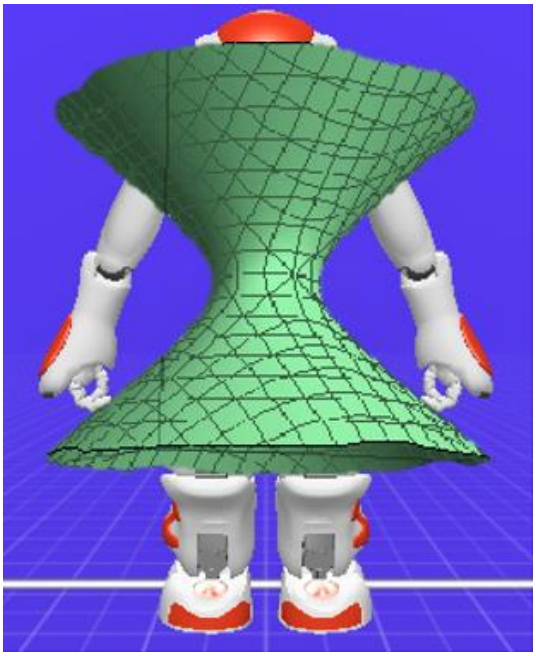
Figura 125. NAO con los hombros desfasados a 90º.



c. Restricción de movimientos de extremidades. Al simular los movimientos, se notó que NAO chocaba con el mismo en algunas ocasiones. Esto se debe a que existen posiciones (x,y,z) que se encuentran dentro de la carcasa de NAO. Por ejemplo, la posición (0,0,0) se encuentra en el centro del "abdomen" de NAO. Tratar de llegar a esta posición resultaría en un choque con la carcasa. Para evitar esto, se modeló la carcasa de NAO como un hiperboloide (ver Figura 126). Los bordes de estos son los límites mínimos a la

que se puede tratar de llegar. Es decir, se restringe el movimiento de las manos de tal modo que estas no pueden entrar al hiperboloide.

Figura 126. Modelado de carcasa como hiperboloide.



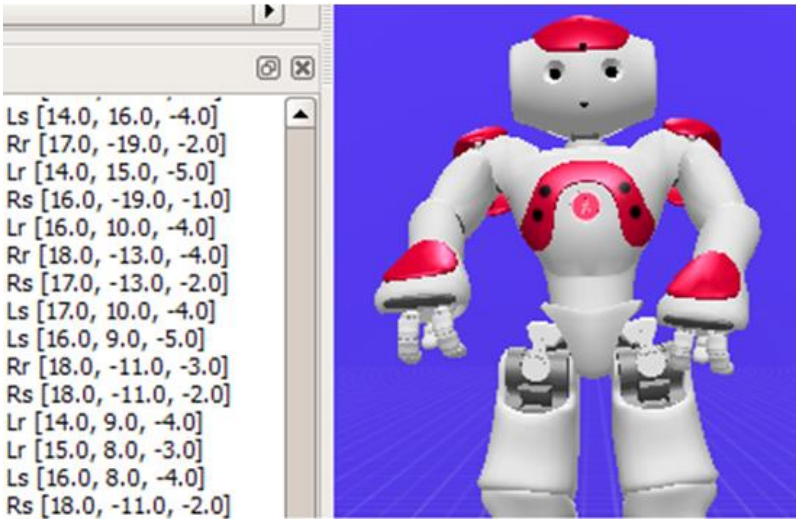
3. Implementación de sistema basado en visión por computadora

a. Movimiento simulado de NAO. El sistema de visión por computadora proveía las posiciones (x,y,z) de las manos del usuario (ya escaladas para la ubicación de las manos de NAO). Se recibían estos datos y se trataba de calcular la cinemática inversa con el método numérico establecido. Debido a que existen eslabones ficticios en la representación física de NAO, era necesario variar el Jacobiano (excluir las columnas ficticias) y establecer los ángulos ficticios en cero. NAO no tiene ángulos ficticios, por lo que enviar estos sería incorrecto. Cuando la comunicación con NAO se estableció, se podía ver que NAO se movía igual que el usuario (en tiempo real). Es decir, replicaba los movimientos del usuario en tiempo real. Para hacer esto, se calculaba la cinemática inversa en MATLAB y se transmitían estos datos inalámbricamente al programa en Python (programa que movía los motores de NAO). Es importante mencionar que al terminar la comunicación (finalizar el programa), se tenía que cerrar el puerto en uso. Si no se hacía esto, no se podría volver al correr el programa (saldría error de que el puerto ya está en uso). Para arreglar esto, se tenía que reiniciar el programa (cerrar y abrir el programa).

Se imprimieron en pantalla las posiciones x, y, z del sistema de visión por computadora y el análisis cinemático. Basándose en estos, se fue variando la constante K para obtener una respuesta rápida, estable y precisa. Al establecer el valor de K, se llegó a un error máximo de 5 cm sobre un eje y 2 cm sobre los otros

dos (ver Figura 127). En esta figura, la primera letra “L” o “R” indica si es del brazo izquierdo o derecho, respectivamente. Las letras que siguen estas, “s” o “r”, indican si la posición es la calculada con cinemática o si es la posición real (la encontrada con sistema de visión por computadora). Estos datos están en centímetros, y como se puede apreciar, la diferencia entre (Lr, Ls) y (Rr, Rs) es menor a 5 cm.

Figura 127. Simulación de NAO y las posiciones reales y calculadas.



b. Movimiento real de NAO. Para comprobar el funcionamiento adecuado de la interfaz y la cinemática inversa, se realizaron dos pruebas: la primera fue la rotación de la cabeza en la cual se le pidió al experimentador que viera a distintas posiciones, tanto en un plano horizontal como vertical, ver Figura 128; la segunda prueba consistió en posicionar los brazos, tanto individual como simultáneamente, en distintas posiciones, ver Figura 129, Figura 130 y Figura 131.

Al momento de hacer las pruebas se observó que NAO realizó los mismos movimientos que el experimentador, aunque los ángulos de sus articulaciones variaban ligeramente. Esto se debe a varias razones, entre las cuales se encuentra la posición de las esferas, ya que estas no se pueden ubicar en el centro de las articulaciones. Otra de las razones es el error causado por la cámara al momento de medir a distintas profundidades. Por último, se observó que el módulo que realiza la cinemática inversa para aplicársela a NAO se desviaba de las posiciones enviadas al robot. [13]

Figura 128. Fotografía de NAO rotando la cabeza, segun la inclinación de la cabeza del experimentador.

Figura 129. Prueba A. levantando ligeramente el brazo derecho, inclinado hacia adelante.

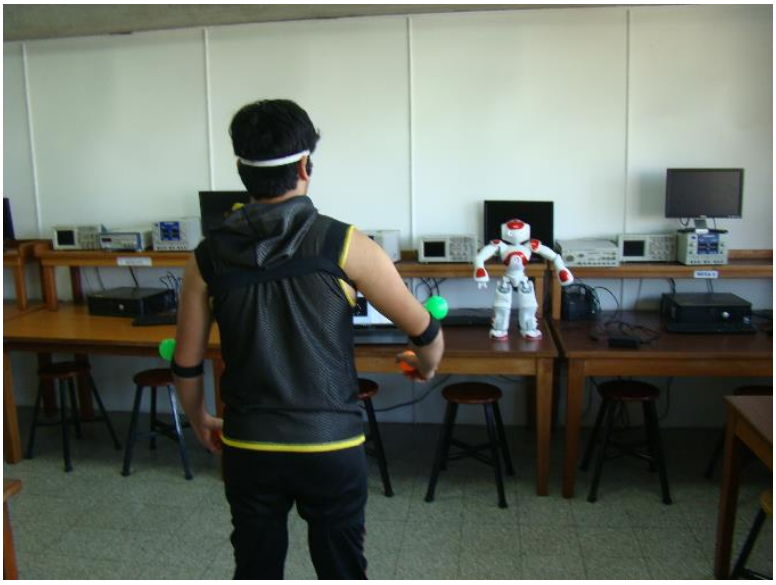


Figura 130. Prueba B. abriendo ambos brazos, inclinados hacia abajo.

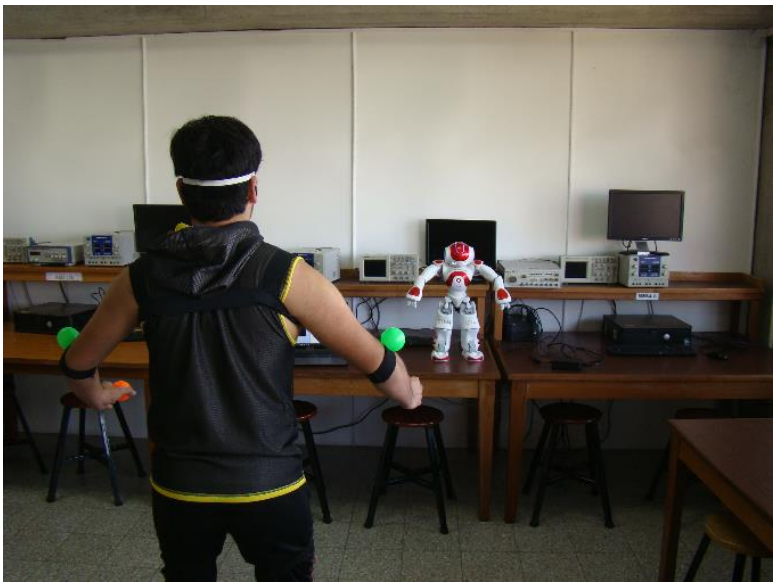
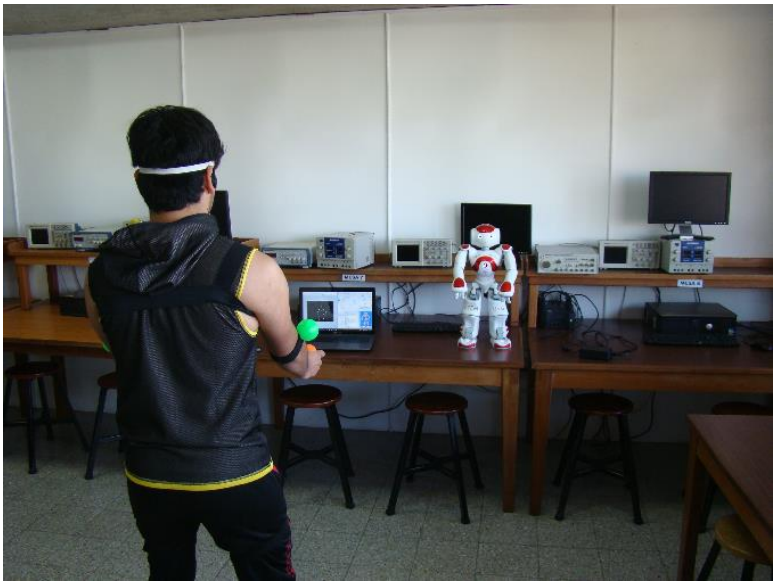


Figura 131. Prueba C. inclinando ambos brazos levemente hacia adelante.



VIII. CONCLUSIONES

1. Se logró la adquisición de señales de EEG en tiempo real utilizando el dispositivo Emotiv EPOC y dos plataformas que permitían la recuperación de sus datos para ser operados en MATLAB: La primera plataforma estaba conformada por los programas de distribución libre BCI2000 y *FieldTrip*, mientras que la segunda correspondía al módulo de obtención de señales de este mismo megaproyecto.

2. Se implementó un programa que permitía el cálculo de los coeficientes de un filtro FIR pasa-banda diseñado utilizando la ventana de Hamming, permitiendo la modificación de la frecuencia de corte y de paso, así como el orden del filtro, para facilitar la evaluación del rendimiento de diferentes filtros al ser utilizados en la detección de la onda P300.

3. Se utilizó un filtro temporal pasa-banda IIR Butterworth de cuarto orden, con una banda de paso entre 1 y 15 Hz, así como un filtro espacial de referencia del promedio común (CAR) para minimizar el ruido introducido por los movimientos del usuario y el resto de su actividad mental.

4. Se halló que, dentro de las cadenas implementadas, aquella que presentó un rendimiento óptimo fue la siguiente: Se aplicó, sobre las señales de EEG, un filtro espacial CAR, seguido de un filtro pasa-banda IIR Butterworth de cuarto orden y una normalización espacial a unidades tipificadas z. Luego, se tomaron las muestras de todos los canales de los 800 ms siguientes a cada estímulo, decimándolos por un factor de dos y promediando aquellas respuestas correspondientes al mismo elemento de la matriz.

5. Debido a la complejidad de las señales de EEG, se decidió utilizar un algoritmo de aprendizaje automático para clasificarlas, siendo el clasificador elegido la máquina de vectores de soporte (SVM) de *kernel* gaussiano, debido a su insensibilidad a la maldición de la dimensión, así como a su capacidad de crear fronteras de decisión no-lineales.

6. Para la clasificación de las señales de EEG en el sistema propuesto, se utilizó un ensamble conformado por tres SVMs, los cuales fueron entrenados y validados con diferentes particiones no disjuntas de un mismo conjunto de datos que correspondían a un mismo usuario.

7. Utilizando las probabilidades de salida del ensamble de SVMs, se podía predecir la letra en el cual el usuario se encontraba concentrado, resultado que era comunicado al módulo de presentación de estímulos para poder brindar una retroalimentación al usuario.

8. Se verificó que la cadena de procesamiento y clasificación propuesta lograba una exactitud promedio 1.5 veces mayor que la lograda por el clasificador incluido con el programa BCI2000.

9. Se realizó una implementación funcional de la cadena de procesamiento y clasificación propuesta para ser utilizada dentro de una aplicación de tiempo real, la cual requiere de un previo entrenamiento del sistema.

10. Se logró obtener, preparar e integrar señales electroencefalográficas con un sistema de inteligencia artificial para desarrollar una aplicación que permita a personas con discapacidades comunicarse por un medio escrito sin necesidad de involucrar actividad motora.

11. Se desarrolló una interfaz gráfica que presenta estimulaciones visuales que generan señales de P300, el cual permite la comunicación cerebro-computadora.

12. Se realizaron pruebas utilizando el gorro de electrodos de Electro-Cap International y el Emotiv EPOC y se determinó que el equipo con mejores resultados para este tipo de aplicación fue el Emotiv EPOC.

13. Con la librería de Matlab © para el Emotiv EPOC, se logró accesar los canales con las señales EEG provenientes del equipo de Emotiv para poder utilizarlas en la interfaz cerebro computadora desarrollado en Matlab.

14. Se logró integrar el programa de procesamiento y clasificación de señales con la interfaz cerebro-computadora de P300 desarrollada en Matlab©.

15. El desempeño de la interfaz cerebro computadora desarrollada supera el desempeño de la interfaz comercial de BCI2000 en un promedio de 27.91%.

16. Se lograron desarrollar interfaces no convencionales basadas en señales provenientes del cuerpo mediante la medición e implementación de señales EMG y visión por computadora.

17. Se lograron evaluar dos métodos de inteligencia artificial para clasificar las señales EMG y posteriormente utilizarlas para controlar un drone (AR Drone 2.0).

18. Se lograron utilizar las mediciones obtenidas del sistema de visión por computadora para controlar un robot humanoide (NAO).

19. Las redes neurales permiten identificar múltiples señales EMG provenientes del antebrazo. La clasificación de señales EMG en función del tiempo permite identificar las variaciones en el comportamiento para músculos independientes. Mientras que la clasificación de señales EMG en función de la frecuencia permite identificar las variaciones en el comportamiento de un conjunto de músculos.

20. Se logró evaluar SVM como modelo de inteligencia artificial para clasificar las señales EMG, sin embargo, este clasificador no se utilizó para enviar comandos ya que tenía pobre estabilidad en las transiciones.

21. Se logró evaluar regresión logística como modelo de inteligencia artificial para clasificar las señales EMG, sin embargo, estas no se convirtieron en comandos ya que no se logró mejorar su rendimiento.

22. Se lograron identificar la cantidad máxima de comandos que se pueden clasificar utilizando SVM manteniendo una buena exactitud (arriba de 90%). En total se lograron cuatro gestos diferentes, con cuatro electrodos secos y con las constantes C y σ iguales a 0.01 y 0.1, respectivamente.

23. No se lograron identificar la cantidad máxima de comandos que se pueden clasificar utilizando regresión logística debido a que este no tenía un buen rendimiento a la hora de clasificar.

24. Se lograron configurar los movimientos para controlar el AR Drone 2.0 a través de una conexión inalámbrica.

25. Las redes neurales permiten identificar cuatro comandos, a partir de señales EMG, sin provocar una clasificación inestable al momento de cada transición en las posiciones de la mano.

26. Por medio de una secuencia de clasificaciones y colocar muestras traslapadas en el clasificador se pueden obtener comandos estables, al momento de realizar transiciones entre cada movimiento.

27. Por medio de cuatro comandos realizados por cada mano, se puede controlar al AR Drone 2.0 para realizar cualquier desplazamiento en tres dimensiones y orientarlo sobre un plano horizontal.

28. Se logró calcular la cinemática inversa de las posiciones obtenidas en el sistema basado en visión por computadora utilizando un método numérico iterativo.

29. Implementando una cámara y marcadores sobre las articulaciones, se pueden identificar las posiciones de cada articulación del cuerpo humano. Implementando un sistema de visión por computadora que identifique colores y contornos.

30. A pesar de que el robot humanoide (NAO) posee proporciones distintas a las del ser humano, se pueden acondicionar las longitudes del cuerpo y brindar posiciones aproximadas de cada articulación. Para ello es deben normalizar las posiciones de las extremidades humanas, sin importar la ubicación del usuario respecto a la cámara.

IX. RECOMENDACIONES

1. Se recomienda realizar pruebas del procesamiento y clasificación de señales propuestas en este trabajo con personas que sufren alguna discapacidad que dificulta o impide su movimiento.

2. Se recomienda probar la cadena de procesamiento propuesta utilizando otros clasificadores, como el SWLDA, para observar si se obtienen ganancias en la exactitud promedio del sistema o no.

3. Se recomienda probar la cadena de procesamiento y clasificación propuesta utilizando un equipo de adquisición de señales de EEG profesional, con el objetivo de observar si se obtienen ganancias en la exactitud promedio del sistema o no.

4. Si se desea continuar esta línea de investigación y desarrollo, se recomienda explorar otras señales provenientes del cerebro, como oscilaciones neuronales y potenciales relacionados a otros estímulos (e.g., a estímulos auditivos), y determinar cuál de estas señales otorga una mejor relación de señal a ruido para proponer interfaces cerebro-computadora más eficaces y precisas.

5. Para asegurar la sincronización entre las iluminaciones y los bloques de muestras obtenidas utilizando la librería de Matlab para el Emotiv EPOC, se recomienda utilizar un mismo temporizador para ambas interrupciones. De esta forma, una depende del otro y no existe desfase entre cada una.

6. Se recomienda obtener el número de muestra directamente del bloque de datos proporcionados de Emotiv en lugar de usar funciones de tiempo de Matlab para calcularlo, ya que este cálculo intermedio puede generar un desfase en tiempo de las muestras tomadas para el clasificador. Es decir, se estaría utilizando otras señales EEG que no corresponden al estímulo actual, que, como consecuencia, afectaría el desempeño final del programa de teclado de P300.

7. Se recomienda utilizar la librería desarrollada en Java para el Emotiv EPOC y comparar el desempeño del programa de teclado de P300, para comprobar cuál obtiene un mejor desempeño.

8. Los modelos de inteligencia artificial clasificaron bien hasta cuatro movimientos. Esto sucedió porque las señales EMG emitidas al efectuar tales movimientos se parecen mucho entre ellas. Para diferenciarlos de una mejor manera y agregar más movimientos, se recomienda utilizar más electrodos.

9. Para una segunda etapa se recomienda utilizar un circuito EMG sin filtrado, debido a que se limita el ancho de banda que se puede analizar. Esto se debe a que se obtienen mejores resultados realizando un análisis de las señales en función de la frecuencia, comparado con uno en función del tiempo.

10. Debido a que se mide las señales EMG provenientes de varios músculos, se recomienda cambiar el diseño de los electrodos a unos que abarquen una mayor sección transversal del antebrazo y así cubrir una mayor cantidad de músculos.

11. Se recomienda realizar un análisis de la energía que presentan las señales EMG, para determinar la fuerza que se está utilizando para cada señal y así poder utilizarlo como otro parámetro de la interfaz.

12. Para evitar que SVM se tardara demasiado en estabilizarse, se podría imprimir la clasificación hasta que se hayan obtenido al menos 5 clasificaciones iguales. Es decir, si pasó de ser la clasificación "1" a la "2", se tendría que esperar a clasificar "2" cuatro veces más. De esta manera, se asegura que la señal clasificada realmente es esta. A pesar de que el tiempo de respuesta aumentará, se ganará estabilidad en la salida y se evitará clasificar erróneamente.

13. Para poder controlar a NAO con mayor facilidad, se recomienda investigar cómo controlar a NAO en MATLAB. Esto para evitar interconectar programas de diferente lenguaje (Python y MATLAB). La desventaja de hacer esto sería que no se podrían simular los movimientos antes de ejecutarlos, sin embargo, se podría simular en Choregraphe y posteriormente implementarlo en MATLAB. Implementarlo de esta manera evitaría problemas de comunicación inalámbrica o problemas con abrir un puerto que ya está en uso (y tener que cerrar el programa para corregir este error).

14. Para una segunda fase se recomienda implementar una segunda cámara, para obtener una mejor percepción de la velocidad y permitir grabar al usuario, aunque cambie su orientación. Esto también permitirá ampliar la región en la que se puede mover NAO (las esferas serán visibles desde dos puntos en vez de solo un punto a la vez).

15. Se recomienda implementar una cámara de alto espectro para mejorar la identificación de objetos o utilizar filtros ópticos para mejorar la identificación de colores.

X. BIBLIOGRAFÍA

[1] Aldebaran Robotics. A-Robots NAO. [En línea]. https://www.aldebaran.com/en/more-about

[2] Aldebaran Robotics. NAO Documentation. [En línea]. http://doc.aldebaran.com/2-1/home_nao.html

[3] Aldebaran Robotics. Robot Software. [En línea]. https://www.aldebaran.com/en/robotics-solutions/robot-software/development

[4] Alderan SoftBank Group. (2006) Aldebaran. [En línea]. https://www.aldebaran.com/en/humanoid-robot/nao-robot

[5] Analog Devices, Inc. *Linear Circuit Design Handbook*, Zumbahlen, Hank, Ed.: Newnes, 2008. [En línea]. http://www.analog.com/library/analogDialogue/archives/43-09/EDCh%208%20filter.pdf

[6] Barea, Rafael. (2014) Universidad de Alcalá, Departamento de Electrónica. [En línea]. http://www.bioingenieria.edu.ar/academica/catedras/bioingenieria2/archivos/apuntes/tema%205%20-%20electroencefalografia.pdf

[7] Blankertz, Benjamin, *et al.* "Predicting BCI performance to study BCI illiteracy," *BMC Neuroscience*, vol. 10, no. 1, pp. 84-88, 2009.

[8] Blankertz, Benjamin; Theresa Vaughan, Gerwin Schalk, y Thilo Hinterberger. (2002) BCI Competition II. [En línea]. http://www.bbci.de/competition/ii/#goals

[9] Blankertz, Benjamin; Theresa Vaughan, Gerwin Schalk, y Thilo Hinterberger. (2012) Programming Tutorial: Working with the FieldTrip Buffer. [En línea]. http://www.bci2000.org/wiki/index.php/Programming_Tutorial:Working_with_the_FieldTrip_buffer

[10] Blankertz, Benjamin; Theresa Vaughan, Gerwin Schalk, y Thilo Hinterberger. (2012, Septiembre) User Tutorial: EEG Measurement Setup. [En línea]. http://www.bci2000.org/wiki/index.php/User_Tutorial:EEG_Measurement_Setup

[11] Cashero, Zachary. "Comparison of EEG Preprocessing Methods to Improve the Classification of P300 Trials," Fort Collins, 2011.

[12] Castañeda, Oscar Fernando. "Interfaces biológicas humano-máquina, Módulo de Procesamiento y Clasificación de Señales para interfaz cerebro computadora," Universidad del Valle de Guatemala, Guatemala, Tesis 2015.

[13] Castillo, Pedro. "INTERFACES BIOLÓGICAS HUMANO-MÁQUINA: Módulo de interpretación y ejecución de comandos para interfaz basada en actividad motora," Guatemala, 2015.

[14] Castillo, Javier. "Interfaces biológicas humano-máquina: Módulo de obtención de señales y desarrollo de aplicación para interfaz cerebro computadora," Guatemala, 2015.

[15] Chang, Ching-Chung y Chih-Jen Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 27, pp. 1-27, 2011.

[16] Chen, Peter. (2008) Biology 3520 Animal Behavior. [En línea]. http://bio3520.nicerweb.com/Locked/chap/ch03/3_11-neuron.jpg

[17] Corke, Peter. *Robotics, Vision and Control: Fundamental algorithms in MATLAB*. Berlin: Springer, 2011.

[18] Criswell, Eleanor. *Cram's Introduction to Surface Electromyography*. Sudbury: Jones and Bartlett, 2011.

[19] De la Fuente, José Marin. (2015, Septiembre) Estados de concentración Alfa, Beta, Theta y Delta. [En línea]. http://www.marindelafuente.com.ar/estados-de-concentracion-alpha-beta-theta-y-delta/

[20] De Vos, Maarten; M. Kroesen, R. Emkes, y S. Debener. "P300 speller BCI with a mobile EEG system: comparison to a traditional amplifier," *Journal of Neural Engineering*, vol. 11, no. 3, pp. 1-8, 2014.

[21] Emotiv. (2014) Epoc. [En línea]. https://emotiv.com/epoc.php

[22] Emotiv Systems. (2014) Emotiv EPOC Neuro Headset. [En línea]. https://emotiv.com/epoc.php

[23] Emotiv Systems. (2014) Emotiv Specifications. [En línea]. https://emotiv.com/product-specs/Emotiv%20EPOC%20Specifications%202014.pdf

[24] Emotiv Systems. (2012) Xavier SDK Wiki. [En línea]. http://wiki.emotiv.com/tiki-index.php?page=include

[25] Escobar, Martha y Hernan Pimienta. *Sistema Nervioso*. Cali: Editorial Universidad del Valle, 2006.

[26] Fazel-Rezai, Reza. "P300-Based Speller Brain-Computer Interface," University of North Dakota, Tesis 2008.

[27] Fazel-Rezai, Reza y Waqas Ahmad. "P300-based Brain-Computer Interface Design," en *Recent Advances in Brain-Computer Interface Systems*. Rijeka: InTech, 2011, pp. 83-98.

[28] Garnier, Stéphane. (2013, Diciembre) Sensor Locations. [En línea]. https://github.com/bschumacher/emokit-old/wiki/Sensor-Locations

[29] Gavin, Paul y Rami Khushaba. (2012, Abril) Matlab Central. [En línea]. http://www.mathworks.com/matlabcentral/fileexchange/36111-emotiveeg-headset-toolbox

[30] Ghasem-Sani, Omid. (2015) Event Related Potentials in Brain-Computer Interfaces. [En línea]. http://www.omidsani.com/

[31] Giménez, Salvador. (2011, Noviembre) Medicina21 Ciencia, medicina, salud y paciente. [En línea]. http://www.medicina21.com/Articulos-V1178-Que_es_un_electroencefalograma.html

[32] Gonsalvez, Craig y John Polich. "P300 amplitude is determined by target-to-target interval," *Psychophysiology*, vol. 39, no. 3, pp. 338-396, 2002.

[33] González, Dulce. (2011) Lóbulos cerebrales.

[34] Graimann, Bernhard; Brendan Allison, y Gert Pfurstscheller. "Brain-Computer Interfaces: A Gentle Introduction," en *Brain-Computer Interfaces*. Berlín: Springer, 2010, pp. 1-27.

[35] Greensted, Andrew. (2010) FIR Filters by Windowing. [En línea]. http://www.labbookpages.co.uk/audio/firWindowing.html

[36] Hauk, Olaf. (2013) Introduction to EEG and MEG. [En línea]. http://imaging.mrc-cbu.cam.ac.uk/meg/IntroEEGMEG#eegrecordings

[37] Hege, Phillip. (2013, Julio) File Exchange. [En línea]. http://www.mathworks.com/matlabcentral/fileexchange/42532-controller-for-ar-drone/content/ARDrone.m

[38] Hsu, Chich-Wei; Ching-Chung Chang, y Chih-Jen Lin. (2010) A Practical Guide to Support Vector Classification. [En línea]. https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[39] Intersil. (1999) An introduction to Digital Filters. [En línea]. http://www.intersil.com/content/dam/Intersil/documents/an96/an9603.pdf

[40] Iowegian, Corporación Internacional. (2015) Decimation. [En línea]. http://dspguru.com/dsp/faqs/multirate/decimation

[41] Iowegian, Corporación Internacional. (2015) IIR Filter Basics. [En línea]. http://dspguru.com/dsp/faqs/iir/basics

[42] Kaper, Matthias; Peter Meinicke, Ulf Grossekathoefer, Thomas Lingner, y Helge Ritter. "BCI Competition 2003 – Data Set IIb: Support Vector Machines for the P300 Speller Paradigm," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1073-1076, 2004.

[43] Kotsiantis, Sotiris. "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, vol. 31, no. 3, pp. 249-268, 2007.

[44] Krusienski, Dean, *et al.* "A comparison of classification techniques for the P300 speller," *Journal of Neural Engineering*, vol. 3, no. 4, pp. 299-305, 2006.

[45] Lacin, Ozgen. (2012) P300-based brain-computer interface. [En línea]. http://www.eee.metu.edu.tr/~ngencer/EE%20517/Lecture%20Notes/Part%20V%20P300-based%20Brain%20Computer%20Interfaces.pdf

[46] Lotte, Fabien; M. Congedo, A. Lécuyer, F. Lamarche, y B. Arnaldi. "A review of classification algorithms for EEG-based brain-computer interfaces," *Journal of Neural Engineering*, vol. 4, no. 2, pp. 24-48, 2007.

[47] Malmivuo, Jaako y Robert Plonsey. *Bioelectromagnetism*. Nueva York: Oxford University Press, 1995.

[48] Manyakov, Nikolay; Nikolay Chumerin, Adrien Combaz, y Marc Van Hulle. (2011) Comparison of Classification Methods for P300 Brain-Computer Interface on Disabled Subjects. [En línea]. http://www.hindawi.com/journals/cin/2011/519868/

[49] Martínez, M.; L. Gómez, A. Serrano, y J. Gómez. (2009) Diseño de filtros FIR. [En línea]. http://ocw.uv.es/ingenieria-y-arquitectura/filtros-digitales/tema_3._diseno_de_filtros_fir.pdf

[50] Mathworks. (2013, Marzo) Matlab Central: Getting Started with NI myDAQ. [En línea]. http://www.mathworks.com/matlabcentral/fileexchange/40749-getting-started-with-ni-mydaq/content/mydaq_gettingstarted.m

[51] Mayo Clinic Foundation for Medical Education and Research. (2015) Functions of the Brain. [En línea]. http://www.mayoclinic.org/functions-of-the-brain/img-20008699

[52] Milsap, Griffin. (2011) Contributions: Emotiv. [En línea]. http://www.bci2000.org/wiki/index.php/Contributions:Emotiv

[53] MINDMEDIA Neuro and Biofeedback Systems. MindMedia. [En línea]. http://biofeedbackspain.es/productos/nuevos-productos/nexus-eeg-cap/

[54] Moore, Keith L.; Arthur F. Dailey, y Anne M. R. Agur. "Músculos del antebrazo," en *Moore- Anatomía con orientación clínica*. Barcelona: Wolters Kluwer Health; Lippincott Wiliams & Wilkins, 2013, pp. 818-823.

[55] Moreno, Ignacio. *Área de Tecnología Electrónica*. Venezuela: Universidad Nacional Experimental del Táchira (UNET), 2010.

[56] Mu Sigma. (2014) Neural Network. [En línea]. http://www.mu-sigma.com/analytics/thought_leadership/cafe-cerebral-neural-network.html

[57] Neuroskills. (2009) Lóbulo Frontal. [En línea]. http://www.neuroskills.com/espanol/lobulo-frontal.php

[58] Neuroskills. (2009) Lóbulo Occipital. [En línea]. http://www.neuroskills.com/espanol/lobulos-occipitales.php

[59] Neuroskills. (2009) Lóbulo Parietal. [En línea]. http://www.neuroskills.com/espanol/lobulos-parietales.php

[60] Neuroskills. (2009) Lóbulo Temporal. [En línea]. http://www.neuroskills.com/espanol/lobulos-temporales.php

[61] Ng, Andrew. (2015) Machine Learning. [En línea]. https://www.coursera.org/learn/machine-learning/home/week/1

[62] Norali, A. N. y M. H. At Som. "Surface Electromyography Signal Processing and Application: A Review," Malaysia, 2009.

[63] Oostenveld, Robert; Pascal Fries, Eric Maris, y Jan Schoffelen. "FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG and Invasive Electrophysiological Data," *Computational Intelligence and Neuroscience*, vol. 2011, no. 1, pp. 1-9, 2011.

[64] Oppenheim, Allan; Ronald Schafer, y John Buck. *Discrete-Time Signal Processing*. Nueva Jersey: Prentice Hall, 1999.

[65] Oxanis Research. (2015) Oxanis Research GMBH. [En línea]. http://www.oxanis.com/about.php

[66] Parrot, S.A. "AR Drone Developer Guide," Paris, 2012.

[67] Parrot, S.A. (2015) Parrot AR Drone 2.0. [En línea]. http://ardrone2.parrot.com/

[68] Pereira, Diana, *et al.* "Effects of inter-stimulus interval duration on the N1 and P2 components of the auditory event-related potential," *International Journal of Psychology*, vol. 94, no. 4, pp. 311-318, 2014.

[69] Picton, Terence; Otavio Lins, y Michael Scherg. "The recording and analysis of event-related potentials," en *Handbook of Neuropshychology*. Ámsterdam: Elsevier, 1995, pp. 3-73.

[70] Punskaya, Elena. (2015) Design of FIR Filters. [En línea]. http://www.vyssotski.ch/BasicsOfInstrumentation/SpikeSorting/Design_of_FIR_Filters.pdf

[71] Rabuñal, Juan R. y Julian Dorado. *Artificial Neural Networks in Real-Life Applications*. España: Idea Group Publichin, 2006.

[72] Rakotomamonjy, Alain y Vincent Guigue. "BCI Competition III: Dataset II - Ensemble SVMs for BCI P300 Speller," *IEEE Transactions in Biomedical Engineering*, vol. 55, no. 3, pp. 1147-1154, 2008.

[73] Rangaswamy, Madhavi y Bernice Porjsz. "From Event-Related Potential to Oscillations," *Alcohol Research & Health*, vol. 31, no. 3, pp. 238-242, 2008.

[74] Repovs, Grega. "Dealing with Noise in EEG Recording and Data Analysis," *Informatica Medica Slovenica*, vol. 15, no. 1, pp. 18-25, 2010.

[75] Roman, Avid. "EEG Signal Processing for BCI Applications," Hal Archives Ouvertes, Cusco , hal-00742211, 2012.

[76] Chowdhury, Rubana H., *et al.* "Surface Electromyography Signal Processing and Classification Techniques," *MDPI AG; Sensors*, pp. 12431-12466, 2013.

[77] Rubia, Francisco J. "Los asombrosos síntomas de la disfunción del lóbulo temporal," *Tendencias21*, Noviembre 2009. [En línea]. http://www.tendencias21.net/neurociencias/Los-asombrosos-sintomas-de-la-disfuncion-del-lobulo-temporal_a15.html

[78] Rubuñal, Juan R. y Julian Dorado. "GA-Based Learning," en *Artificial Neural Networks in Real-Life Applications*. United Kingdom: Idea Group Publishing, 2006, pp. 319-320.

[79] Ruiz, A. F.; F. J. Brunetti, E. Rocon, A. Forner-Cordero, y J. L. Pons. "Adquisición y procesado de información EMG en el modelado de sistemas biológicos," Madrid.

[80] Rutkove, Seward. (2010, Abril) PhysioNet. [En línea]. http://physionet.org/physiobank/database/emgdb/

[81] Schalk, Gerwin. (2015) BCI2000. [En línea]. http://www.schalklab.org/research/bci2000

[82] Studio, Serendip. (2012, Septiembre) Brain Structures and their Functions. [En línea]. http://serendip.brynmawr.edu/bb/kinser/Structure1.html

[83] Smith, Julius. (2007) Forward-Backward Filtering. [En línea]. https://ccrma.stanford.edu/~jos/fp/Forward_Backward_Filtering.html

[84] Spong, Mark, *et al. Robot Modeling and Control*. New York: Wiley, 2006.

[85] Sterratt, David, *et al. Principles of Computational Modelling in Neuroscience*. Cambridge: Cambridge University Press, 2011.

[86] Szeliski, Richard. "What is computer vision?," en *Computer Vision algorithms and applications*. New York: Springer, 2011, pp. 3-9.

[87] Tecayehuatl, Eric. (2009, Diciembre) Emotiv EPOC: Práctico lector de ondas cerebrales. [En línea]. https://www.fayerwayer.com/2009/12/emotiv-epoc-practico-lector-de-ondas-cerebrales/

[88] Teplan, M. "Fundamentals of EEG Measurement," *Measurement Science Review*, vol. 2, no. 2, pp. 1-11, 2002.

[89] Teplan, Michael. "Fundaments of EEG Measurement," *Measurement Science Review*, vol. 2, 2002.

[90] Thakor, Nitish y David Sherman. "EEG Signal Processing: Theory and Applications," en *Neural Engineering*. Nueva York: Springer, 2013, pp. 259-303.

[91] The University of Chicago Medical Center. (2015) The University of Chicago Medical Center. [En línea]. http://www.uchospitals.edu/online-library/content=S03866

[92] Thevenet, Daniel. (2008) CEEIBS. [En línea]. http://www.nib.fmed.edu.uy/ceeibs/Clase_09.pdf

[93] Tinku Acharya, Ajoy K. Ray. "Perceptually Uniform Color Space," en *Image Processing: Principles and Applications*. New Jersey: Wiley Interscience, 2005, pp. 41-45.

[94] Universidad de Barcelona. (2006) Psicología de la Percepción Visual. [En línea]. http://www.ub.edu/pa1/node/130

[95] Universidad del País Vasco. (2008) Resumen de las características de los filtros analógicos Butterworth y Chebyshev. [En línea]. http://www.ehu.eus/Procesadodesenales/tema6/tx33.html

[96] University of Maryland Medical Center. (2015) University of Maryland Medical Center. [En línea]. http://umm.edu/health/medical/spanishency/articles/electroencefalograma

[97] Valdeavellano, Mario; Luis Reina, Gerardo Martínez, y Hans Bahnsen. "Megaproyecto ANIMA: Métodos no convencionales de interfaz de control de robots a través de electroencefalografía y la electrooculografía," Guatemala, 2009.

[98] Valdeavellano, Mario Roberto; Luis Fernando Reina, Gerardo Estuardo Martínez, y Hans Juergen Bahnsen. "Métodos no convencionales de interfaz en el control de robots a través de la electroencefalografía y electrooculografía," Universidad del Valle de Guatemala, Guatemala, Tesis 2009.

[99] Zanak Jamal, Muhammad. "Signal Acquisition Using Surface EMG and Circuit Design Considerations for Robotic Prosthesis," en *Computational Intelligence in Electromyography Analysis-A Perspective on Current Applications and Future Challenges*.: Creative Commons, 2012, pp. 428-448.