UNIVERSIDAD DE LAS AMÉRICAS PUEBLA

Escuela de ingeniería

Departamento de Computación, Electrónica y Mecatrónica

Uso de la varianza y el clasificador SVM para la detección de potenciales P300 y propuesta para diseño de Interfaz Cerebro-Computadora

Tesis que, para completar los requisitos del Programa de Honores presenta la estudiante

Diana Pérez Mosqueda

162131

Ingeniería Biomédica

Directora de tesis: Dra. Rocío Salazar Varas

San Andrés Cholula, Puebla. Otoño 2022

Hoja de firmas

Tesis que, para completar los requisitos del Programa de Honores presenta el estudiante Diana Pérez Mosqueda con ID 162131

Directora de Tesis

_____

Dra. Rocío Salazar Varas

Presidente de Tesis

_____

Dr. Jorge Rodríguez Asomoza

Secretario de Tesis

_____

Dr. Juan Horacio Espinoza Rodríguez

Índice

Resumen

Las Interfaces Cerebro-Computadora (BCI's, del inglés Brain-Computer Interfaces) son sistemas que permiten interpretar la información cerebral para traducir la intención del usuario, y mediante el procedimiento adecuado, tomarla como un control a un dispositivo. Una de sus aplicaciones se encuentra en los deletreadores basados en potenciales P300. Estos potenciales se presentan instantes posteriores a que el sujeto recibe un estímulo (audible o visual) trascendente.

En el presente trabajo se muestra, en primera instancia, la comparación de diferentes características para la detección del potencial P300 en combinación con una máquina de soporte vectorial (del inglés Support-Vector Machine, SVM). Los resultados reflejan que la varianza de la señal es una buena característica para esta aplicación. Para evaluar el desempeño del sistema se empleó una base de datos pública, logrando un porcentaje de clasificaciones correctas del 98.4 %.

Posteriormente se presenta una propuesta de interfaz cerebro computadora basada en potenciales P300, desarrollada en MATLAB.

Palabras clave: BCI, EEG, Potenciales Evocados, P300, Clasificación SVM.

Tabla de abreviaturas

EEG: Electroencefalografía.

BCI: Interfaz Cerebro-Computadora (del inglés Brain-Computer Interface).

SVM: Máquinas de Vectores de Soporte (del inglés Support-Vector Machine).

P300: Potencial Evocado 300 ms posteriores a un estímulo.

ERP: Potenciales Relacionados a Eventos (del inglés Event-Related Potentials)

1. Introducción

Las interfaces cerebro-computadora (BCI's por sus siglas en inglés) presentan avances sumamente importantes para aumentar las capacidades de interacción que tenemos los seres humanos con el mundo que nos rodea. Especialmente es una promesa de apoyo que resulta relevante para personas con algún tipo de discapacidad (Millán et al., 2010).

El poder distinguir patrones en la actividad cerebral de un usuario es un reto que aprovecha las herramientas e innovación para resultar en una tecnología asistida y de utilidad. El presente proyecto de investigación retoma metodologías y conceptos de la literatura en el área de estudio para proponer las bases para un propio BCI utilizando diversas herramientas de programación y adquisición.

En este trabajo específicamente se presenta una adaptación del paradigma de un deletreador que funciona con potenciales evocados P300 cuyo funcionamiento y metodología son explicados en la sección del Marco Teórico.

Este proyecto se divide en dos secciones principales: la primera consta de la clasificación de señales de EEG (electroencefalograma) con potenciales evocados P300 utilizando una base de datos pública, mientras que la segunda parte presenta una propuesta para el desarrollo y diseño de una interfaz gráfica para clasificar potenciales evocados.

2. Justificación

En las últimas dos décadas, la expectativa de vida de la población ha incrementado significativamente, 7.4% (The World Bank, 2019) lo que aumenta la cantidad de personas que resultan susceptibles a algún tipo de discapacidad o enfermedad. Algunas de estas condiciones afectan a los canales neuromusculares, dañando la comunicación entre el cerebro y el control que el cuerpo tiene con el ambiente externo. Por mencionar varios ejemplos se identifican la esclerosis lateral amiotrófica, esclerosis múltiple, distrofia muscular, accidente cerebrovascular del tronco encefálico, entre otros (Brennan et al., 2014).

Esto presenta a la tecnología la oportunidad de aportar como asistencia a todos aquellos que necesitan una mejora en su calidad de vida y brindar mayor independencia, mediante la innovación en el campo de la salud.

Las interfaces cerebro-computadora ofrecen a los usuarios una manera de comunicar y controlar aplicaciones y dispositivos sin la intervención de nervios periféricos o músculos. Debido a esto, la investigación realizada se ha centrado en construir proyectos de utilidad para esta aplicación, asegurándose de garantizar una buena calidad de la señal adquirida con equipo de bajo costo y sencillo de utilizar.

Así mismo, se espera que facilite la tarea de aquellas personas encargadas de cuidar a personas con alguna de estas condiciones. Potencialmente, podría conocerse información en el cerebro del usuario que no puede expresar de manera convencional (Nicolas y Gómez, 2012).

Hoy en día, existen distintos prototipos que permiten a los usuarios controlar sillas de ruedas, teclados e incluso juegos computacionales. Sin embargo, se espera que el prospecto principal de esta tecnología pueda mejorar la vida de incontables personas que sufren de algún tipo de discapacidad ya que, de acuerdo con datos del Banco Mundial, en 2021 aproximadamente el 15 % de la población mundial (1000 millones de personas) vive con una discapacidad (Banco Mundial, 2021). Se proyecta que puedan aprovecharse combinándolas con tecnologías asistivas ya existentes (Millán et al., 2010).

3. Objetivos

Calcular y evaluar diferentes características cuantitativas de las señales de EEG reportadas previamente en la literatura, para la detección de potenciales P300 y proponer una interfaz cerebro computadora simple, basada en estos potenciales.

3.1 Objetivos específicos

- Evaluación de diferentes características previamente reportadas en la literatura para la detección de potenciales P300.
- Desarrollo de un clasificador SVM en Matlab.
- Evaluación del sistema de clasificación en condiciones *offline* y pseudo-*online* utilizando una base de datos pública.
- Diseño e implementación de una interfaz gráfica básica para la detección de potenciales P300.

4. Marco Teórico

En esta sección se presentan las bases teóricas que fundamentan el proceso realizado durante el proyecto.

4.1 Electroencefalografía (EEG)

El EEG es una técnica no invasiva que registra la actividad cerebral mediante electrodos distribuidos en el cuero cabelludo. Específicamente, esta técnica mide la actividad eléctrica causada por el flujo de corrientes eléctricas que ocurren en las dendritas de las neuronas durante la excitación sináptica (Nicolas y Gómez, 2012) (Saccá et al., 2018).

Los electrodos mencionados son colocados utilizando como referencia el Sistema Internacional 10-20 que ha sido estandarizado por la *American Electroencephalographic Society*. Este sistema utiliza dos puntos de referencia en la cabeza para definir la locación de los electrodos (uno por arriba de la nariz a nivel de los ojos y otro en el inión en la base del cráneo). Los planos transversos y medianos dividen al cráneo desde dichos puntos y la colocación de los electrodos se determina marcando estos planos en intervalos de 10% y 20%. La Figura 1 muestra dichos intervalos y la división de regiones con los respectivos nombres de los electrodos. La letra correspondiente a cada locación refiere a una región cerebral, específicamente A al lóbulo de la oreja, C la región central, P_g la nasofaríngea, P la parietal, F la frontal, F_p la polar frontal y O el área occipital (Nicolas y Gómez, 2012).

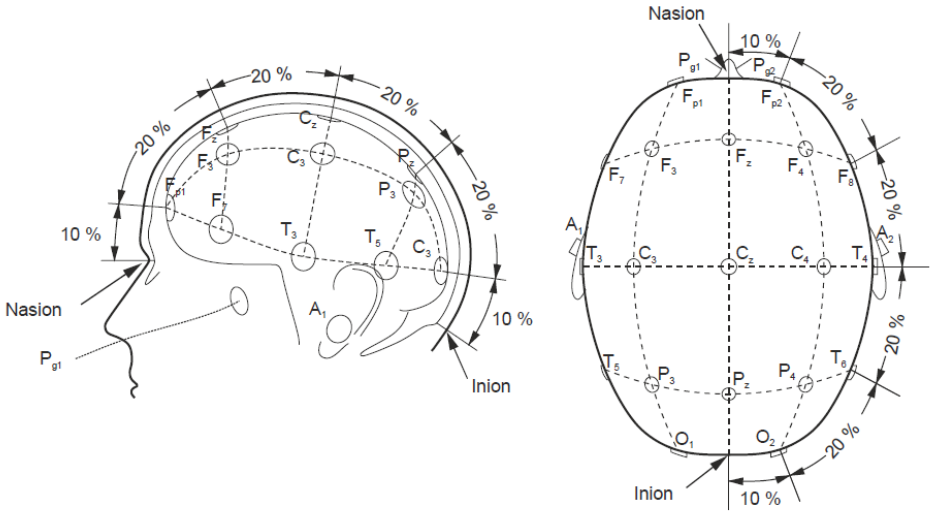

Figura 1. Sistema Internacional 10-20 (Nicolas y Gómez, 2012)

Debido a su modalidad y sus características no invasivas, no dolorosas y sin efectos secundarios, suele ser utilizado como un método de diagnóstico clínico con interpretaciones precisas para enfermedades cerebrales tales como la epilepsia, autismo, pérdida de memoria, Alzheimer entre otras. (Kumar y Bhuvaneswari, 2012).

De la misma manera, estas señales de EEG registradas pueden ser clasificadas para identificar distintos estados o estímulos basándose en sus frecuencias y otros parámetros. Este tipo de análisis resultan de gran importancia para la investigación y entendimiento de diversas características del complejo comportamiento de la estructura del cerebro. Sin embargo, también presentan dificultades en cuanto a la presencia de ruido debido a la basta actividad cerebral y, por lo tanto, se presenta el reto de una detección automática de los estados cerebrales de interés (Saccá et al., 2018) (Kumar y Bhuvaneswari, 2012).

4.2 Interfaces Cerebro-Computadora (BCI)

Una de las más innovadoras y prometedoras maneras de utilizar las señales de EEG es en las interfaces cerebro-computadora (BCI por sus siglas en inglés), las cuales son sistemas que permiten tomar la información cerebral como un control a un equipo o dispositivo. Es decir, puede interpretar la intención de control por parte de un usuario mediante el procedimiento adecuado.

La Figura 2 ilustra de manera generalizada el funcionamiento básico de una BCI, notando que están conformadas por diferentes etapas desde la intención del usuario medido en sus señales de EEG, procesado y clasificado hasta obtener un resultado satisfactorio en la comunicación de ésta.

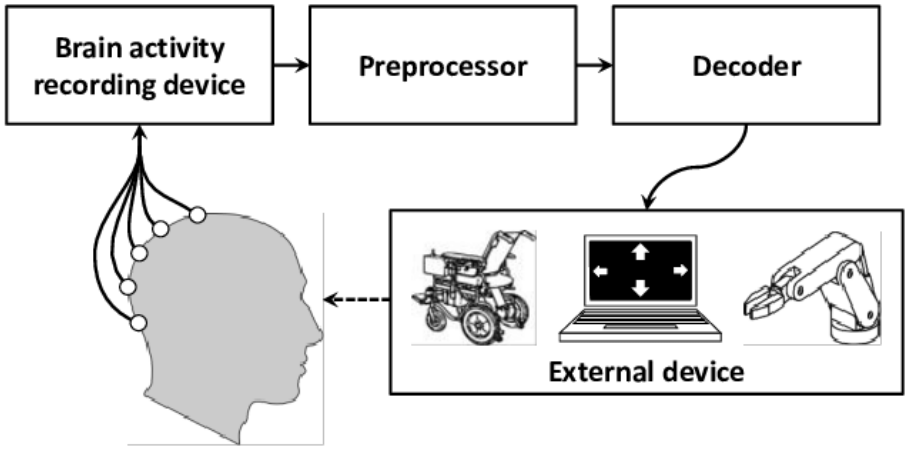

Figura 2. Funcionamiento general de sistemas BCI (Mora et al., 2014)

De acuerdo con Brennan et al. (2014), el potencial de las BCI se basa principalmente en que no requiere de movimiento físico por parte del usuario y puede ser categorizado como sistemas invasivos (aquellos que requieren la colocación de arreglos de microelectrodos

directamente en el córtex para derivar la intención del usuario) o no invasivos (aquellos que no requieren ningún tipo de intervención quirúrgica para colocar los electrodos y obtener la actividad cerebral). En la mayoría de los casos se prefiere el uso de los últimos para su aplicación en BCI para evitar los posibles riesgos clínicos y dificultades técnicas significativas.

En el ámbito médico pueden tener un correcto funcionamiento determinando las intenciones del usuario y logrando la comunicación con diversos dispositivos, sin embargo, existen aún bastantes aplicaciones que presentan limitaciones importantes en cuanto a la complejidad, configuración y precisión, de acuerdo con los mismos autores mencionados.

Retomando las diferentes etapas que conforman a las BCI, Van Gerven et al. (2009) sintetizan con mayor detalle el ciclo que siguen estos sistemas en la Figura 3.

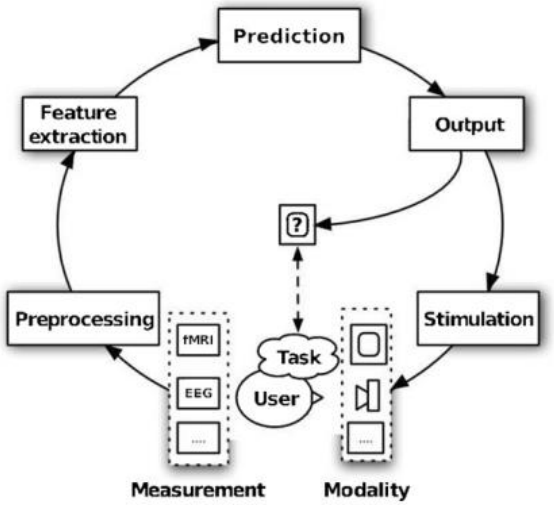

Figura 3. Ciclo de etapas de BCI. El usuario realiza una tarea mental con un estímulo la cual es medida, preprocesada, se extraen las características y se utiliza algún método o algoritmo para predecir un comando, el cual es la salida del sistema (Van Gerven et al., 2009).

Comienza con el usuario realizando alguna tarea mental mientras recibe un estímulo y utilizando sensores es registrada la actividad cerebral. Posteriormente las señales obtenidas son preprocesadas adecuadamente para obtener después las características relevantes o de interés asociadas a la tarea mental desarrollada. El resultado del sistema actúa como una señal de salida que controla un dispositivo externo. Los autores lo consideran un ciclo debido a que el usuario es capaz de percibir el resultado y puede evaluar el correcto funcionamiento del dispositivo, sin embargo, este puede depender de la habilidad del usuario y puede presentarse en diversas modalidades. Durante varias iteraciones del ciclo, el usuario y la computadora pueden tener un tipo de aprendizaje y entrenamiento, lo cual puede resultar en una mejora del funcionamiento del sistema. La Figura 4 presenta una tabla realizada por Fang et al. (2021) en donde sintetizan aquellos algoritmos comúnmente utilizados en la bibliografía para la sección del preprocesamiento, extracción de características y la clasificación.

Procedure	Algorithm
Preprocessing	Temporal Filter
	Spatial Filter
	Frequency Filter
	Time-Frequency Filter
	Time-Spatial Filter
Feature Extraction	Fast Fourier Transform（FFT）
	Wavelt Transform（WT）
	Auto Regression(AR)
	Independent Component Analysis(ICA)
	Principal Component Analysis(PCA)
Feature Classification	Bayes Decision
	Support Vector Machine(SVM)
	Linear Discriminant Analysis(LDA)
	Artificial Neural Network(ANN)
	Common Spatial Patterns(CSP)

Figura 4. Procedimientos y algoritmos utilizados en BCI, tomado de Fang et al. (2021)

A continuación, se presenta una descripción individual de las secciones generales que conforman una BCI, fundamentada en el trabajo de Nicolas y Gómez (2012).

4.2.1 Adquisición

Utilizando dispositivos de diagnóstico, se registran las señales de EEG. Autores como Fang et al. (2021) resaltan la importancia de trabajar con un paradigma experimental de adquisición apropiadamente diseñado para poder obtener señales de alta calidad y como consecuencia extraer de la manera más optima la información necesaria por parte del usuario.

4.2.2 Preprocesamiento

En esta etapa se eliminan los artefactos asociados con la electromiografía (EMG) o electrooculografía (EOG), además de reducir el ruido. De manera general, mediante diferentes herramientas (filtros, ventanas de tiempo, etc.), la señal se prepara lo mejor posible para poder ser utilizada en etapas posteriores dentro del sistema.

4.2.3 Extracción de características

Esta parte identifica la información discriminativa en las señales obtenidas. Una vez que es obtenida, la señal es mapeada en un vector que contiene todas las características discriminantes y efectivas de las señales. Este paso en el proceso es una tarea con complicaciones debido a que las señales contienen la información de diversas actividades cerebrales que se traslapan en tiempo y espacio además de ser señales no estacionarias.

El vector de características debe de ser de una dimensión baja para reducir la complejidad de la extracción, pero sin dejar de lado la información relevante.

4.2.4 Clasificación y Control

Durante esta etapa se realiza la clasificación de las señales tomando en cuenta los vectores de características generados. La elección de las características adecuadas es esencial para

poder realizar el reconocimiento de patrones de manera efectiva y obtener la intención del usuario.

Finalmente, el control de la interfaz consiste en utilizar las señales clasificadas como comandos significativos hacia algún dispositivo externo.

4.3 Potencial Evocado P300 y Paradigma *Odd-ball*

P300 es un potencial cognitivo relacionado a un evento (ERP por sus siglas en inglés) el cual es emitido a nivel cerebral por alguna toma de decisión o inicio de una respuesta. Se encuentra estrechamente asociado con el paradigma experimental *Odd-ball*, en el cual un sujeto reacciona a las incidencias presentadas en un grupo de estímulos. Estas incidencias ocurren ocasionalmente dentro de una serie de estímulos comunes, por lo tanto, se considera al estímulo "normal" como *Non Target* y al "anormal" como *Target* (Vareka y Mautner, 2015).

Como consecuencia de esta reacción, la onda P300 aparece en la zona parieto-central de la cabeza y se muestra en la señal EEG como un componente positivo con una latencia entre 250 y 300 ms y su amplitud es más grande después del estímulo de *Target*. En la Figura 5 se ejemplifica el aspecto de dicho potencial evocado con y sin estímulo (Farwell y Donchin, 1988) (Vareka y Mautner, 2015).

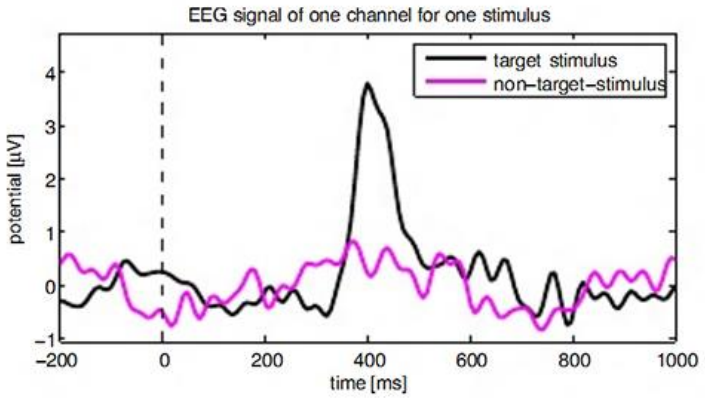

Figura 5. Ilustración comparativa entre estímulo P300 consecuencia de Target vs. Estímulo Non Target (Microsiervos, 2014).

Retomando el trabajo de Farwell y Donchin (1988), existe amplia documentación sobre la elicitación del P300 directamente relacionada con la relevancia de los eventos e inversamente con la probabilidad del estímulo. Por lo tanto, la elicitación depende de la habilidad del sujeto para discriminar los eventos y asignarlos a la categoría correcta. Es por esto por lo que se recomienda que el usuario simplemente mantenga un conteo mental de estos estímulos sin ser necesario algún otro tipo de acción motora o verbal.

4.3.1 Deletreador

Las BCI's proveen de un canal de comunicación directo que traducen la actividad cerebral y, por lo tanto, uno de los temas más relevantes en su estudio y desarrollo es el de los deletreadores para la comunicación. Pueden utilizarse diversos tipos de señales de control como los ERP (potenciales relacionados a eventos) o SSVEP (potenciales evocados visualmente en estado estacionario). La Figura 6 ilustra la configuración más común del

equipo de EEG junto con la BCI donde se observan los estímulos que son mostrados al usuario.

Este deletreador es sumamente utilizado debido a que su función recae en los ERP inducidos por el paradigma *Odd-ball* previamente descrito. La Figura 7 ejemplifica el tipo de interfaz que es mostrada en un deletreador de este tipo donde la iluminación de los caracteres induce al estímulo deseado. La metodología más común utilizada para este tipo de aplicación es descrita más adelante (Hwang et al., 2017).

Figura 6. Ejemplo de usuario utilizando un deletreador P300 (Bitbrain, 2020)

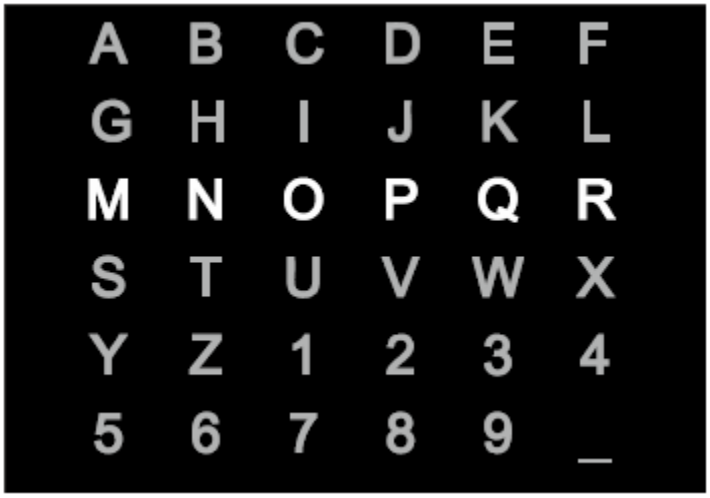

Figura 7. Ejemplificación de interfaz para deletreador (Nicolas y Gómez, 2012)

4.4 Algoritmos utilizados para clasificación de P300

Autores como Fang et al. (2021) han sintetizado aquellos algoritmos comúnmente utilizados en trabajos para la clasificación de la señal característica P300, identificando la decisión Bayesiana, LDA (análisis linear discriminante), SVM (máquinas de vectores de soporte), CSP (patrón espacial común) y ANN (redes neuronales artificiales).

En su trabajo realizan una comparación entre las ventajas y desventajas en su clasificación al utilizar cada uno de ellos, resaltando las ANN como de los algoritmos mayormente elegidos actualmente para clasificar señales de EEG ya que permiten la reducción de errores cambiando los pesos sinápticos entre las neuronas de una red. Sin embargo, a su vez también reconocen la habilidad de generalización que brindan los algoritmos de SVM y su adecuación para diversos problemas de clasificación de señales.

Sobre este último algoritmo resaltan la relevancia de considerar las diferencias entre escenarios y sujetos como factores que pueden afectar a los resultados favorables en la clasificación.

4.4.1 Máquinas de vectores de soporte (SVM)

Las máquinas de vectores de soporte (SVM por sus siglas en inglés) son un modelo de aprendizaje automático supervisado que es capaz de realizar clasificaciones binarias de manera generalizada y para el caso más simple de dicha clasificación se genera un hiperplano encargado de separar las clases (Kaper y Ritter, 2004).

Dicho hiperplano es ilustrado en la Figura 8, donde puede observarse el ejemplo de una gráfica de dispersión marcando geométricamente la división entre ambos grupos.

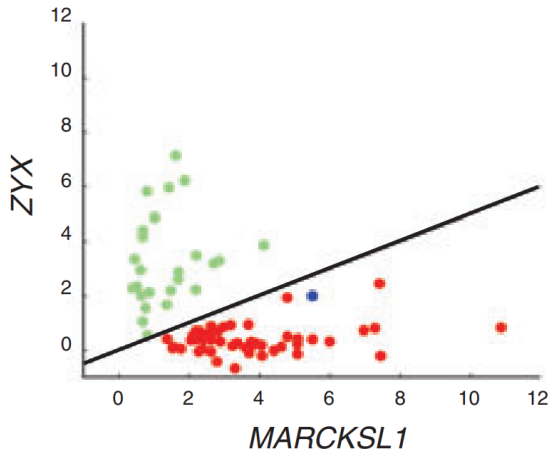

Figura 8. Ejemplificación de gráfico de dispersión (Noble,2006).

Considerando lo anterior, un clasificador de aprendizaje automático crea un hiperplano tomando l ejemplos de vectores de entrenamiento (x_i) con las etiquetas correspondientes:

$$y_i: (x1, y1), \dots, (xl, yl) \in R^N \qquad (1)$$

El hiperplano es descrito por el vector w y por el *bias* b, y cada que se proyecta un nuevo vector (x) de un conjunto de prueba en el vector w, se revela la clase asignada de acuerdo con el signo de la proyección.

$$f(x) = w \cdot x + b \qquad (2)$$

Específicamente, un SVM se encarga de maximizar el margen γ (Figura 9) para poder encontrar aquel hiperplano óptimo que brinde la mejor capacidad de generalización. Los vectores de soporte son aquellos que se encuentran en el margen del hiperplano y son esenciales para la descripción del clasificador generado. En la misma figura se muestran encerrados en un círculo (Kaper y Ritter, 2004).

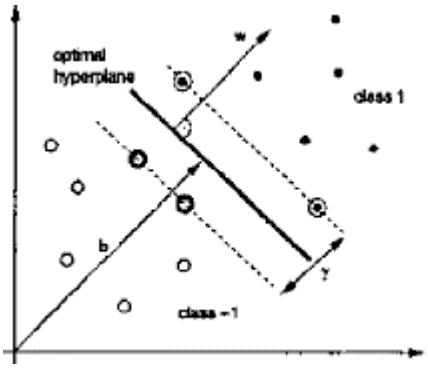

Figura 9. Maximización de margen del hiperplano (Kaper y Ritter, 2004).

Es importante recalcar que el tratar los objetos o datos a clasificar como puntos en un espacio de alta dimensión y encontrar una línea que los separe no es exclusivo del SVM. Sin embargo, se diferencia de otros clasificadores también basados en hiperplanos en virtud de cómo se selecciona el hiperplano (Noble, 2006).

4.5 Metodologías previas

Anteriores estudios hicieron uso de potenciales evocados P300 para el desarrollo de sistemas de Interfaces Cerebro-Computadora. Gran parte de dichas investigaciones se han centrado en el uso de estos para la función de deletreadores visuales que son útiles para personas con algún tipo de discapacidad.

Tomando en cuenta diferentes autores tales como Kaper y Ritter (2004), Martínez, et al. (2017), Silvoni et al. (2009) y Hwang et al. (2017), el funcionamiento de estos sistemas de manera general comprende una matriz de caracteres aleatorios dentro de la cual se iluminan distintas filas y columnas al azar. El caracter deseado por el usuario debe de ser observado y contar la cantidad de veces que se ilumina para poder concentrarse en él. Se espera que el potencial P300 resulte de dicha iluminación y utilizando un procesamiento y análisis de datos se puede identificar la intersección de la iluminación de una fila y una columna para indicar el símbolo que fijó el usuario. Generalmente es necesario repetir los registros para poder obtener datos suficientes para su clasificación debido a la baja relación Señal-Ruido (SNR).

Chaurasiya et al. en 2016 emplearon SVM como algoritmo de clasificación para detectar el potencial P300 en un deletreador cuya tarea es encontrar el caracter deseado por el usuario.

Retoman la ventaja que ofrece debido a su mejor capacidad de generalización comparada con otros algoritmos, al igual que es seleccionado por la clasificación binaria necesaria para resolver la clasificación del deletreador.

5. Metodología

El proyecto fue desarrollado en Matlab debido a que utiliza lenguaje de alto nivel, además de tener librerías que permiten el procesamiento de las señales de EEG, tiene facilidad de conexión con distintos dispositivos útiles para el desarrollo de proyectos tales como Arduino y es posible generar modelos de máquinas de vectores de soporte.

Como se mencionó previamente, este proyecto se divide en dos partes. La primera parte consiste en la evaluación de diferentes características y la clasificación de señales de EEG para la detección de potenciales evocados P300. En la segunda parte se presenta una propuesta de diseño de una BCI sencilla, basada en potenciales P300, así como su implementación.

5.1 Evaluación de características y clasificación de señales

La Figura 10 muestra un diagrama de flujo que resume el procedimiento desarrollado para la extracción de características y la clasificación de las señales de EEG para la detección de potenciales P300.

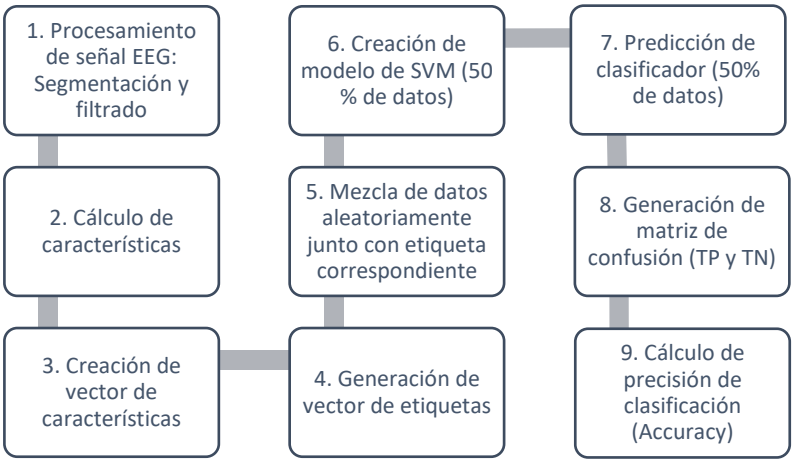

Figura 10. Metodología para generación de clasificador.

5.1.1 Base de datos

Para el desarrollo y la evaluación de la primera parte del proyecto se utilizó la base de datos *P300 speller paradigm* proporcionado por *el Wadsworth Center* del Departamento de Salud del Estado de Nueva York (por Jonathan R. Wolpaw, Gerwin Schalk, Dean Krusienski) y que fueron grabados con la interfaz BCI2000 (Blankertz, 2004).

Específicamente se adquirieron señales (filtradas entre 0.1 y 60 Hz y digitalizadas a 240 Hz) de dos sujetos en cinco sesiones, durante las cuales la persona se concentraba en una serie de caracteres. Se utilizaron los 64 electrodos designados en la Figura 11, cuyos números corresponden al canal registrado. Para cada iteración de caracter se les mostró una matriz durante 2.5 s iluminada a la misma intensidad y después cada fila y columna de la matriz fue intensificada de manera aleatoria por 100 ms. Posteriormente la matriz volvió a iluminarse uniformemente por 75 ms.

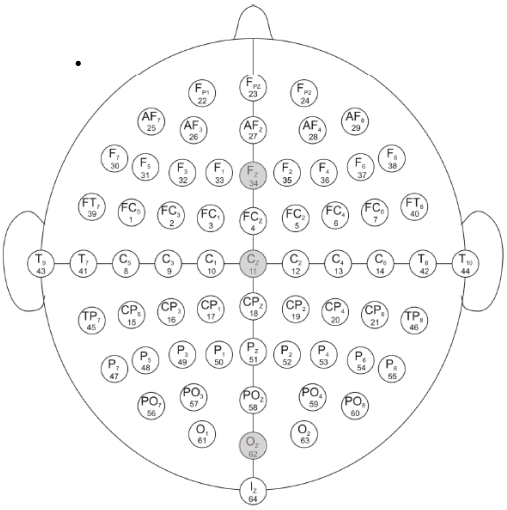

Figura 11. Diagrama de cabeza con electrodos, indicando con sombreado aquellos seleccionados. Adaptado de Blankertz (2004).

Las intensificaciones de las filas y columnas se realizaron en bloques de 12 repeticiones en 5 iteraciones. Se obtuvieron un total de 180 intensificaciones para cada iteración de caracteres. Entre cada iteración de caracteres se dio un periodo de 2.5 s con la matriz uniformemente iluminada indicando al usuario que el caracter fue completado y puede enfocarse en el siguiente para poder armar una palabra.

Esta información se convirtió en archivos de Matlab conteniendo la adquisición de 85 caracteres para cada sujeto. Por lo tanto, la señal de interés contiene las dimensiones de 85 registros * 7794 muestras * 64 canales.

5.1.2 Extracción de características

Para el análisis de la señal se tomaron segmentos de un segundo de registro posterior a la intensificación de la fila o columna. Esto genera una matriz de dimensiones 85 registros *

240 muestras * 64 canales. Se formaron dos matrices correspondientes a las dos categorías posibles: *Target*, si el caracter visualizado por el usuario es intensificado, y *Non Target*, cuando el caracter visualizado por el usuario no es intensificado.

A fin de trabajar con un número reducido de electrodos, se analizó la topología de la señal de EEG sin estímulo y con estímulo, buscando conservar los tres electrodos donde se aprecie mayor diferencia entre las dos clases (*Target/Non Target*).

Una vez que la señal ha sido segmentada en ventanas de 1 s (240 muestras) y cada registro es segmentado en ventanas de 300 ms (72 muestras). Este proceso se realizó con el propósito de analizar el segmento de la señal con mayor detalle tal y como fue propuesto por Vareka y Mautner (2015).

Posteriormente la señal fue filtrada utilizando filtros Butterworth con una frecuencia de corte inferior de 2 Hz y una de corte superior de 30 Hz para eliminar información no deseado de la señal.

Tras realizar este proceso, para cada una de las ventanas se calculan los distintos parámetros para caracterizar la señal, de cada uno de los tres canales seleccionados, para obtener el vector de características. Específicamente Polinomios para Ajuste polinomial de curvas, Dimensión fractal, Coherencia, Varianza y Correlación.

5.1.2.1 Polinomios

Para extraer la primera característica (Polinomios para el Ajuste polinomial de curvas), se utilizó la función *polyfit* de Matlab , la cual calcula los coeficientes del polinomio $p(x)$ de grado n que se ajusten o encajen de manera más apropiada a los datos de la señal (Ecuación

3). Se generaron puntos equidistantes en la señal, a los cuales se le ajustó a un polinomio de grado 3. Los coeficientes de p están en potencias descendentes y la longitud de p es n+1 tal y como expresa la ecuación (Matlab, 2022):

$$p(x) = p_1 x^n + p_1 x^{n-1} + \cdots + p_n x + p_{n+1} \qquad (3)$$

5.1.2.2 Dimensión fractal

La dimensión fractal fue obtenida utilizando la función *hfd* (Monge, 2022) (Al-Kadi, O, 2022), la cual utiliza el algoritmo de Higuchi para calcularla. De manera general este algoritmo estima la longitud media de una curva utilizando como medida un segmento de *k* muestras *L* (Ecuación 4) (Polychronaki et al. 2010). Para el cálculo se utilizó en cada uno de los canales tanto para *Target* y *Non Target* y se consideró 15 como el valor máximo para *k.*

$$L(k) = \frac{1}{k}\sum_{m=1}^{k} L_m\,(k) \qquad (4)$$

5.1.2.3 Coherencia

La coherencia fue obtenida utilizando *mscoherence*, (*Coherence Estimate of Two Sequences*, 2022) la cual se comporta en función de la frecuencia y obtiene valores entre 0 y 1, los cuales representan la correspondencia entre dos señales (x , y) en cada frecuencia. Fueron comparadas las señales en los tres canales tanto para *Target* como para *Non Target*. Como tal, la coherencia de magnitud cuadrada (Ecuación 5) es una función de las densidades espectrales de potencia de $P_{xx}(f)$ y $P_{yy}(f)$:

$$C_{xx}(f) = \frac{|P_{xx}(f)|^2}{P_{xx}(f)P_{yy}(f)} \qquad (5)$$

5.1.2.4 Varianza

Para el cálculo de la varianza, se utilizó la función *var* en Matlab, la cual es obtenida de todos los elementos del arreglo y de forma predeterminada se normaliza a n-1 siendo n la cantidad de observaciones (*Variance of Matrix,* 2022). La fórmula utilizada para obtener este parámetro se muestra en la Ecuación 6.

$$\sigma^2 = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n-1} \qquad (6)$$

5.1.2.5 Correlación

Finalmente, la correlación es obtenida como un coeficiente de dos variables (A y B) que mide su dependencia linear, se calculó con *corrcoef* (*Random columns of Matrix,* 2022) comparando las señales en pares utilizando las combinaciones posibles para *Target* y para *Non Target.* Se calcula el coeficiente de correlación de Pearson utilizando la siguiente fórmula donde N son las observaciones escalares*:*

$$\rho(A,B) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{A_i - \mu_A}{\sigma_A}\right)\left(\frac{B_i - \mu_B}{\sigma_B}\right) \qquad (7)$$

Así pues, para cada registro se obtuvo un vector de características ***y*** conteniendo los parámetros estimados en cada una de las ventanas, para cada uno de los canales, eg. al utilizar la varianza ***y***= $[\sigma^2_{ch1v1}, \sigma^2_{ch1v2}, \sigma^2_{ch1v3}, \sigma^2_{ch2v1}, \sigma^2_{ch2v2}, \sigma^2_{ch2v3}, \sigma^2_{ch3v1}, \sigma^2_{ch3v2}, \ \sigma^2_{ch3v3}]$.

A fin de preparar los datos para la etapa de clasificación, los vectores de características se agruparon en una matriz de características colocando primero aquellos correspondientes a las señales donde hay estímulo (*Target*) seguidos de los vectores que corresponden a señales sin estímulo (*Non Target*).

5.1.3 Desarrollo del clasificador *offline*

Para el desarrollo del clasificador (Anexo A) se consideró el siguiente procedimiento:

5.1.4 Clasificación

A la matriz de características creada, de acuerdo con el caso (*Target* – 1, *NonTarget* - 0) se agregó una columna más de etiquetas (labels) cuya función es identificar con un valor númerico el tipo de señal.

Se realizó una validación cruzada aleatoria de 30 iteraciones, reacomodando la matriz con las etiquetas en los grupos de datos para la creación y entrenamiento del clasificador. Posteriormente, se indexaron la mitad de los datos para ser utilizados en la creación del modelo (considerada Y real), especificamente se utilizó un modelo de máquina de vectrores de soporte (SVM) en Matlab que permite el entrenamiento para una clasificación binaria de un conjunto de datos (Matworks, 2022).

Posterior a la generación del modelo se indexó el resto de los datos para clasificarlos, considerando el resultado de dicha clasificación como Y estimada. Mediante la creación de una matriz de confusión (Figura 12) para la Y real y la Y estimada, se obtuvo el cálculo promedio de la precisión (*Accuracy*) del clasificador en las 30 iteraciones.

Finalmente, para evaluar los resultados, la precisión fue calculada utilizando la fórmula presentada en la Ecuación 8, donde se toman en cuenta las detecciones Verdaderas positivas (*TP*) y Verdaderas negativas (*TN*).

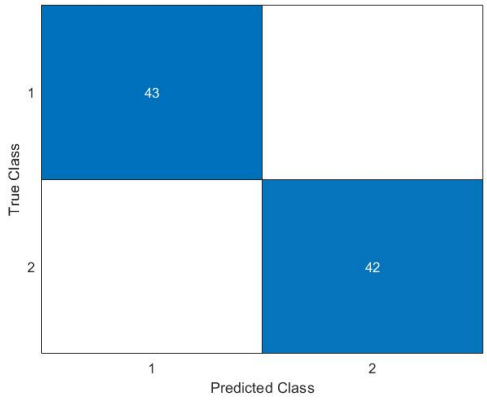

Figura 12. Matriz de confusión generada por una iteración

$$Precisión = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \qquad (8)$$

5.1.5 Desarrollo de clasificador pseudo-*online*

De la misma manera, se desarrolló un código que permite la clasificación de la señal de manera pseudo-*online* (Anexo G), y cuyo objetivo es el de realizar el proceso simulando una clasificación en línea, es decir, sin hacer la segmentación previa de los datos (ventanas de 1 s posterior a la intensificación de la fila/columna). Tomando en cuenta la misma base de datos utilizada en el clasificador *offline*, se trabajaron con los siguientes vectores:

- *Signal*: Contiene la señal completa registrada de EEG.
- *Flashing:* Contiene los instantes donde hubo una intensificación o estímulo.
- *StimulusType*: Contiene la información sobre sí el usuario veía el estímulo o no.

Cuando en el vector de *Flashing* se encuentra un estímulo (cambio de 0 a 1) a través de una resta, en el vector *Signal* para los tres canales seleccionados se crean ventanas de tiempo de 1 segundo y a su vez son divididas en ventanas de 300 ms. Para cada ventana se obtiene la varianza y se forma la matriz de características. La selección de la varianza como parámetro para la clasificación se debe a los resultados posteriormente reportados, donde se identifica como aquel que presenta una mayor precisión.

Utilizando el mismo modelo ya generado y entrenado del clasificador *offline*, se clasifica el vector de características obtenidas y se calcula la Y estimada.

La Y real es obtenida del vector *StimulusType*, el cual fue segmentado de la misma manera que la señal. Para tener los estímulos en un vector de ceros y unos, en cada una de las ventanas de tiempo se realizó una suma de la señal, si el resultado de esta es distinto de cero, se considera que en esa ventana existe un estímulo y es guardado en un vector. Se creó un vector formado de valores de ceros del tamaño de la señal y se sustituyeron por valores de unos en aquellos índices donde fue encontrado el estímulo.

Posteriormente, con la matriz de confusión fue calculada la precisión de la clasificación de la misma manera en la que fue calculada con el clasificador *offline.*

5.2 Propuesta de interfaz

La segunda sección del proyecto está conformada por la propuesta del desarrollo de una BCI que, por medio del clasificador diseñado, pretende identificar las señales de EEG del usuario para poder enviar un comando hacia un dispositivo actuador (Arduino). El alcance de esta

propuesta llega hasta la sección del entrenamiento del modelo, sin embargo, se presenta la posible metodología y material para completar la transmisión del comando al Arduino

. En la Figura 13 se muestra un diagrama a bloques de la propuesta.

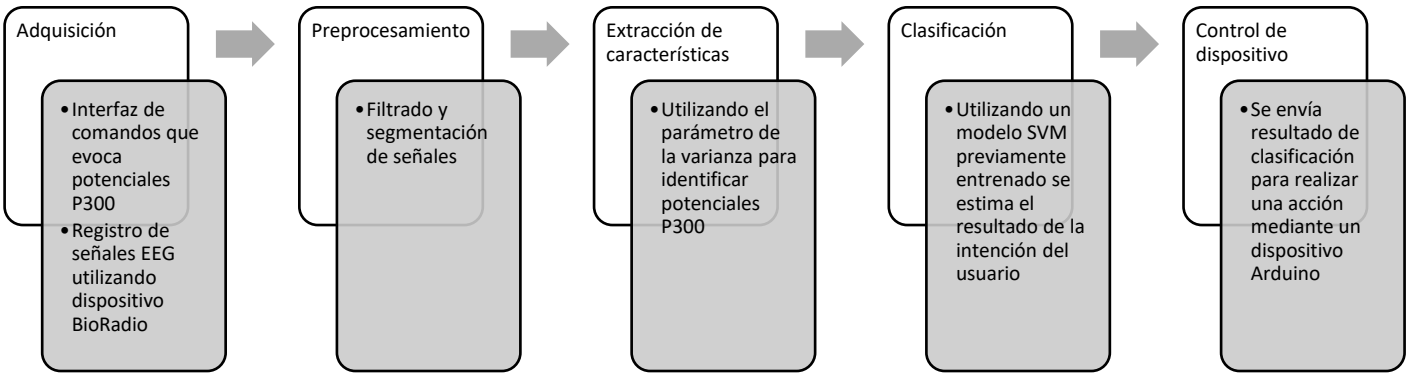

Figura 13. Diagrama a bloques de la BCI propuesta

5.2.1 Diseño de interfaz

La propuesta de diseño es una interfaz creada en Matlab (Anexo H) que sea clara para el usuario, se le mostrarán tres cuadrados de diferente color, distribuidos de manera uniforme en la pantalla. Cada cuadrado estará asociado a una acción (por ejemplo, encendido/apagado de luces, activación/desactivación de aire acondicionado, encendido/apagado de música), por lo que el usuario deberá centrar su atención en el cuadrado vinculado a la acción de su interés.

Se presenta al usuario con los tres tipos de estímulo de manera aleatoria con un tiempo de cambio entre ellos aleatorio ente 0.3 a 0.8 segundos.

5.2.2 Adquisición

Se propone el uso del dispositivo *BioRadio*™ para llevar a cabo la adquisición de las señales de EEG. Este dispositivo es un sistema portable sencillo de operar que puede ser configurado para transmitir distintos tipos de señales en tiempo real, así como almacenarlas en una computadora utilizando diferentes frecuencias de muestreo, para este proyecto se seleccionó una frecuencia de muestreo de 500 Hz (Great Lakes NeuroTechnologies, 2020).

Otra característica importante sobre este dispositivo es su conexión con Matlab, en donde es posible guardar aquellas variables de interés de la señal para poder ser procesadas. Específicamente los índices de tiempo en los cuales se registra un estímulo y cuál es el estímulo registrado.

El objetivo de la adquisición de datos es poder entrenar al modelo utilizando cada uno de los colores o casos de clasificación. Por lo tanto, en cada ronda de 20 registros el usuario se fijará en un color en específico hasta completar los 3 casos. Es recomendable hacer previamente una prueba para la correcta adquisición de la señal antes de comenzar con el registro.

Con base en lo observado para la clasificación offline, se propone el uso de 3 canales (Fz, Cz y Oz), una tierra y una referencia utilizando electrodos secos, adaptados para poder ser ubicados como se ilustra en la Figura 14. La tierra es colocada en la zona ventral de la muñeca de la mano derecha debajo del pulgar, mientras que la referencia se coloca en el lóbulo de la oreja derecha. En el dispositivo de adquisición y estos canales se conectan correspondiendo a la Figura 15.

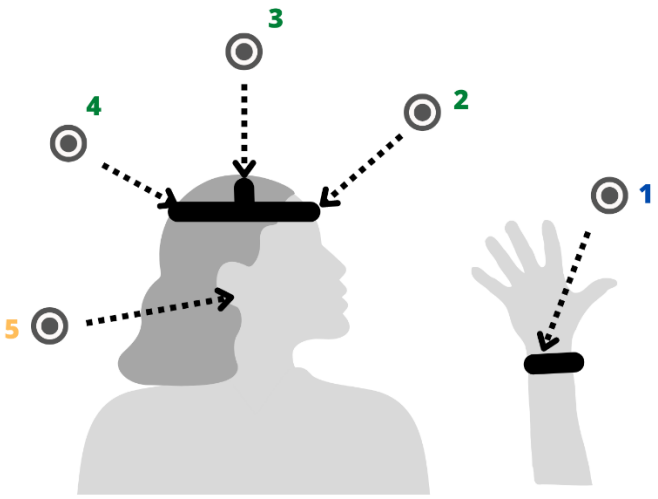

Figura 14. Propuesta de la colocación de electrodos donde: 1. Tierra. 2. Electrodo Fz. 3. Electrodo Cz. 4. Electrodo Oz. 5. Electrodo de referencia

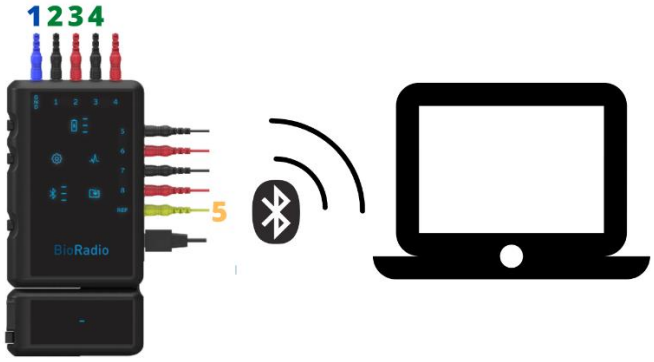

Figura 15. Conexión con dispositivo BioRadio (Great Lakes NeuroTechnologies, 2020) y conexión con computadora.

5.2.3 Programa de entrenamiento

Para poder entrenar al sistema de manera *offline* (Anexo I) se propone utilizar los tres casos de observación del mismo sujeto obtenidos en una sola sesión de adquisición, con el fin de evitar un sesgo en los datos registrados.

Este programa propuesto se basa completamente en el desarrollado para la clasificación *offline* y permite la separación de las señales en vectores distintos para trabajarlos por *Target*, *NonTarget* y toma como parámetro de clasificación la varianza debido a los resultados de clasificación, mostrados en la sección de Resultados.

Si el programa de clasificación identifica un potencial, se activará mediante el Arduino, una señal de control dependiendo de qué cuadro esté observando el sujeto (Anexo J).

6. Resultados

6.1 Evaluación de características y detección del P300

Como se mencionó en la metodología, una parte crucial es la selección de electrodos que se encuentran en aquellas zonas donde espacialmente se observa mayor cambio en la actividad cerebral después de aproximadamente 300 ms del estímulo, por lo que para su elección se mapeó dicha actividad en distintas latencias en el cuero cabelludo utilizando aquellas señales que contienen un estímulo P300 (Figura 16) y aquellas donde no se presenta (Figura 17). Dichas gráficas e ilustraciones se generaron utilizando EEGlab en Matlab (Delorme y Makeig, 2004). Como puede observarse, al comparar la distribución espacial de la actividad cerebral a los 300ms (distribución enmarcada en rojo en las Figuras 16 y 17), en el caso de la clase Target se observa mayor actividad para la zona central en comparación a lo que se observa cuando no hay estímulo. Esta observación llevó a que se seleccionaron los electrodos Cz, Fz y Oz, numerados 11, 34 y 62 respectivamente en la designación mostrada en la Figura 11.

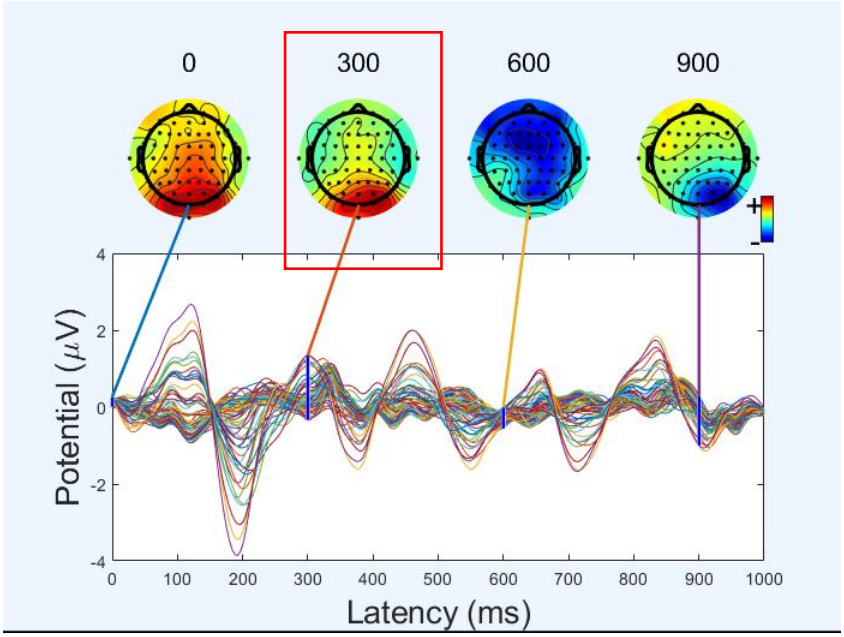

Figura 16. Mapeo de actividad de Target en distintas latencias.

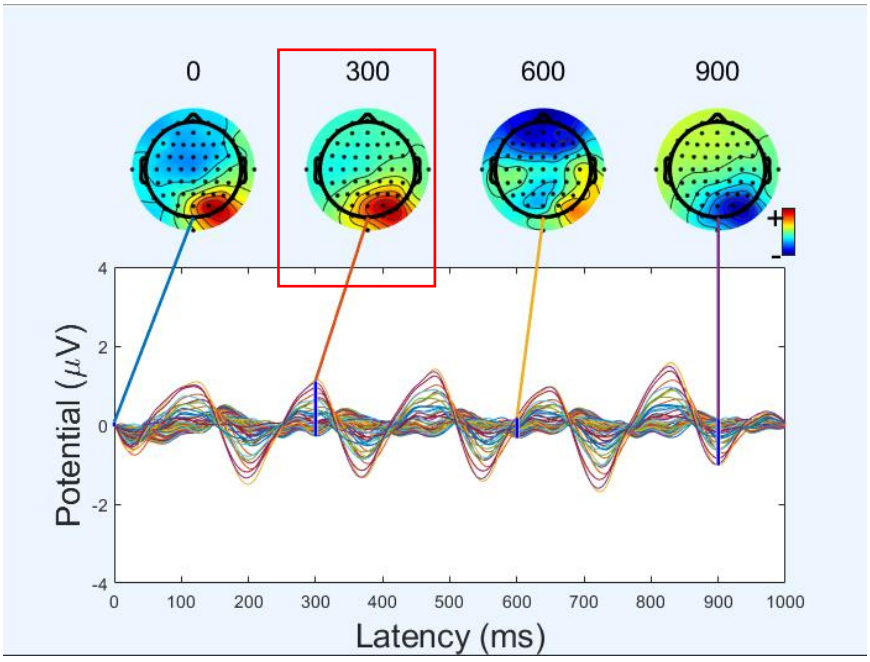

Figura 17. Mapeo de actividad de Non Target en distintas latencias.

En la Figura 18 se muestran una gráfica de dispersión de los datos en el espacio de características, en la cual se observa la división entre las clases. Los puntos rojos corresponden a la señal cuando contiene el potencial P300 y los cyan corresponden a la señal de EEG sin potencial. Este diagrama fue generado usando la varianza como característica. El eje X y el eje Y corresponden a los valores de la varianza obtenida de los canales Fz y Cz, respectivamente.

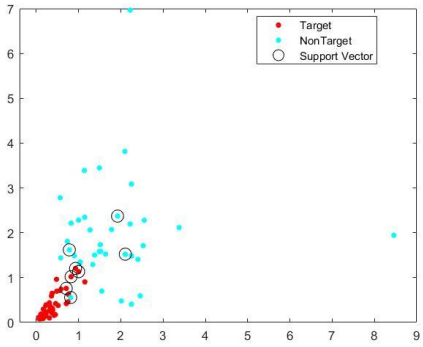

Figura 18. Gráfica de dispersión de datos de modelo generado.

Tal y como se indica en la metodología, la precisión de la clasificación fue calculada utilizando los datos obtenidos de la matriz de confusión para cada una de las iteraciones. Dicho cálculo se obtuvo utilizando la fórmula en la Ecuación 8 (Vareka y Mautner, 2015) donde *tp* son las detecciones verdaderas positivas, *tn* verdaderas negativas, *fp* falsas positivas y *fn* falsas negativas.

Este cálculo se realizó para las 30 iteraciones para cada parámetro y se obtuvo un promedio de clasificación mostrado en la Tabla 1. Se observa que la precisión de clasificación

más alta se obtuvo utilizando la varianza. Clasificadores desarrollados en los Anexos B, C, D y E.

Parámetros	1. Polinomios para Ajuste polinomial de curvas	2. Dimensión fractal	3. Coherencia	4. Varianza	5. Correlación
Electrodos considerados	Cz	Cz, Fz, Oz	Cz, Oz	Cz, Fz, Oz	Cz, Fz, Oz
Clasificación promedio (%)	93.2 ± 5.72e-02	82.3 ± 0.18	95.8 ± 4.78e-02	98.4 ± 9.24e-03	66.0 ± 8.20e-02

Tabla 1. Resultados de clasificación offline de parámetros

Para probar estadísticamente la diferencia entre cada una de las medias resultantes de la clasificación de parámetros se realizó en Matlab una prueba bilateral para la hipótesis nula de que dos valores provienen de una misma distribución (Anexo F).

Considerando la hipótesis nula como que todas las medias son iguales donde μ1 es la media de clasificación para el parámetro 1, μ2 del parámetro 2 y así sucesivamente:

$$H0= \mu1= \mu2= \mu3= \mu4= \mu5$$

Se evaluó la diferencia del resultado obtenido con la varianza (μ4) con el resto de las medias. En todos los casos se pudo rechazar la hipótesis nula, lo que permite concluir que existe una diferencia significativa entre los resultados obtenidos.

El interés de resaltar la diferencia entre la media del parámetro de la varianza con el resto es debido al objetivo de utilizar aquel que brinde mejores resultados en la clasificación del modelo entrenado.

La precisión obtenida como resultado de la clasificación pseudo-*online* utilizando la varianza como parámetro, fue 62% el cual contrasta con los resultados de la clasificación *offline*, esto debido a las diferencias entre los valores de los segmentos de la señal utilizada. Para la clasificación *offline*, se utilizaron ventanas de señal previamente recortadas conociendo dónde se aplicó el estímulo, mientras que para la pseudo-*online* se utilizó la señal sin ser previamente segmentada, es decir, utilizando todo el registro a lo largo del tiempo.

El diseño de la interfaz que se muestra al usuario para generar los estímulos durante la adquisición descrita en la metodología se muestra en la Figura 19. Una consideración tomada para el diseño fue el ajustar los cuadros de manera que se encontraran lo más separados posibles y que abarcaran poco espacio para evitar la distracción visual del usuario y facilitar su enfoque en el color de interés.

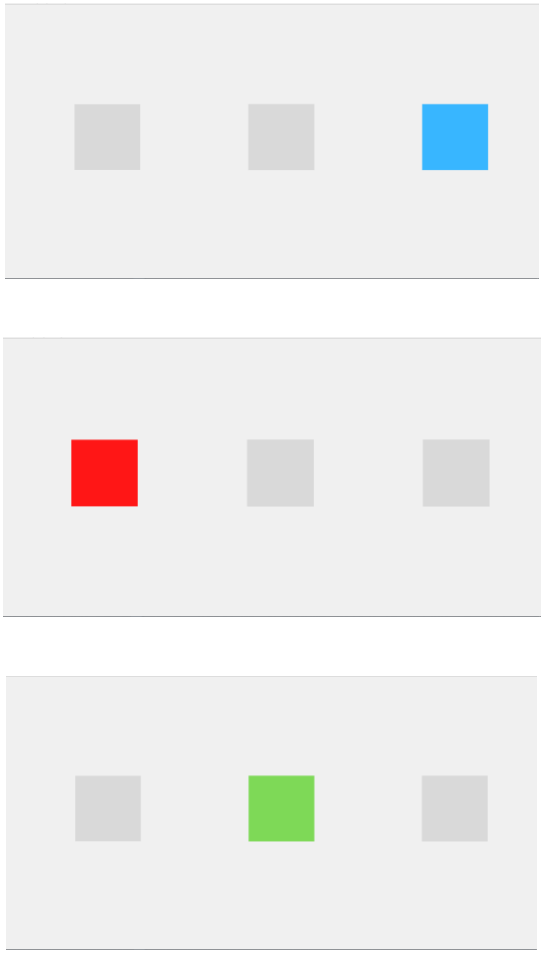

Figura 19. Propuesta de diseño de interfaz con tres posibles comandos (rojo, verde y azul).

7. Conclusiones y Recomendaciones

El presente trabajo ha permitido la aplicación de los métodos de preprocesamiento de las señales de EEG que a su vez fue clasificada utilizando un algoritmo SVM. De la misma manera, se ha realizado una revisión de los conceptos teóricos del tema sobre los cuales se basa la metodología presentada.

Dentro de los resultados de clasificación obtenidos utilizando cinco distintos parámetros, tres de ellos indican un rendimiento mayor al 90% con el clasificador *offline*, sobresaliendo la varianza como el mejor clasificado (98.4%).

La propuesta para la interfaz gráfica es lograda presentando el diseño, posible programa de adquisición, de clasificación y la manera en la cual la salida o decisión es enviada a un dispositivo (Arduino). También se presenta la posible colocación de los electrodos y su conexión al dispositivo de adquisición.

Por lo tanto, los objetivos generales y específicos planteados para el proyecto de investigación han sido satisfactoriamente cumplidos. Como perspectiva futura a este trabajo de investigación se sugiere hacer un mínimo de 60 iteraciones para cada registro con el objetivo de evitar sobreajuste de los datos en la clasificación. También se recomienda que el arreglo de electrodos utilizados para la adquisición permita que sea realicen con el menor ruido posible para poder llevar a cabo el procesamiento posterior en la interfaz, al igual que enfocarse en qué objetos del mundo exterior pueden ser controlados que resulten de utilidad al usuario (encender las luces del hogar, por ejemplo). Así mismo, es recomendable realizar la selección de electrodos dependiendo del sujeto ya que en este trabajo fueron elegidos basándose en la base de datos pública.

8. Referencias

Al-Kadi, O. (2022). Fractal Dimension (https://www.mathworks.com/matlabcentral/fileexchange/44951-fractal-dimension), MATLAB Central File Exchange.

Arnaud Delorme (2022). EEGLAB (https://github.com/sccn/eeglab), *GitHub*.

Banco Mundial. (2021). *La inclusión de las personas con discapacidad, clave para el desarrollo sostenible de América Latina y el Caribe*. World Bank; Banco Mundial. https://www.bancomundial.org/es/news/press-release/2021/12/02/la-inclusion-de-las-personas-con-discapacidad-clave-para-el-desarrollo-sostenible-de-america-latina-y-el-caribe#:~:text=WASHINGTON%2C%202%20de%20diciembre%20de,y%20sociales%20existentes%20y%20la

Bitbrain. (2020). *Interfaces Cerebro-Computador*. Aplicaciones. https://www.bitbrain.com/es/aplicaciones/interfaces-cerebro-computador

Blankertz, B. (2004). *BCI Competition III*. Bbci.de. https://bbci.de/competition/iii/

Great Lakes NeuroTechnologies. (2020). *BioRadio*. https://www.glneurotech.com/products/bioradio/

Brennan, C., McCullagh, P. J., Lightbody, G., and Galway, L. (2014). The BCI as a pervasive technology: a research plan. *In Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth '14). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)*, Brussels, BEL, 252–255. https://doi-org.udlap.idm.oclc.org/10.4108/icst.pervasivehealth.2014.255755

Chaurasiya, R. K., Londhe, N. D., Ghosh, S. (2016). A Novel Weighted Edit Distance-Based Spelling Correction Approach for Improving the Reliability of Devanagari Script-Based P300 Speller System. *IEEE Access*, vol. 4, pp. 8184-8198, doi: 10.1109/ACCESS.2016.2614494

Delorme A, Makeig S. (2004). EEGLAB: an open-source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *J Neurosci Methods*. 2004 Mar 15;134(1):9-21. doi: 10.1016/j.jneumeth.2003.10.009. PMID: 15102499.

Fang, T. et al. (2021). Recent Advances of P300 Speller Paradigms and Algorithms. *2021 9th International Winter Conference on Brain-Computer Interface (BCI)*, 2021, pp. 1-6, doi: 10.1109/BCI51272.2021.9385369.

Farwell, L., Donchin, E. (1988). Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology,* 70(6), 510–523. doi:10.1016/0013-4694(88)90149-6

Hwang, J. -Y., Lee, M. -H., Lee, S. -W. (2017). A brain-computer interface speller using peripheral stimulus-based SSVEP and P300. *5th International Winter Conference on Brain-Computer Interface (BCI)*. pp. 77-78, doi: 10.1109/IWW-BCI.2017.7858164.

Kaper, M., Ritter, H. (2004). Progress in P300-based brain-computer interfacing. *IEEE International Workshop on Biomedical Circuits and Systems*. pp. S3/5/INV-S3/59, doi: 10.1109/BIOCAS.2004.1454154.

Kumar, J. S., Bhuvaneswari, P. (2012). Analysis of Electroencephalography (EEG) Signals and Its Categorization–A Study. *Procedia Engineering*, 38, 2525–2536. doi:10.1016/j.proeng.2012.06.298

Martínez, V., Gómez, J., Álvarez, D., Hornero, R. (2017). An Asynchronous P300-Based Brain-Computer Interface Web Browser for Severely Disabled People. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 8, pp. 1332-1342, Aug. 2017, doi: 10.1109/TNSRE.2016.2623381.

Matworks. (2022). *Train SVM Classifier*. Mathworks.com. https://la.mathworks.com/help/stats/fitcsvm.html

Mathworks. (2022). *Variance of Matrix*. Mathworks.com. https://la.mathworks.com/help/matlab/ref/var.html#buo1t_w

Mathworks. (2022). *Coherence Estimate of Two Sequences*. Mathworks.com. https://la.mathworks.com/help/signal/ref/mscohere.html#bvi4l0m-2

Mathworks. (2022). *Random Columns of Matrix*. Mathworks.com. https://la.mathworks.com/help/matlab/ref/corrcoef.html#bunkanr

Matlab. (2022). *Ajustar un polinomio a la función trigonométrica*. Mathworks.com. https://la.mathworks.com/help/matlab/ref/polyfit.html

Microsiervos. (2014). ¿Funciona científicamente la “prueba de la verdad” P300? *Microsiervos.* https://www.microsiervos.com/archivo/ciencia/miguel-carcano-y-la-prueba-de-la-verdad-con-el-potencial-evocado-cognitivo-p300.html

Millán, J. D. R., Rupp, R., Müller-Putz, G., Murray-Smith, R., Giugliemma, C., Tangermann, M., ... y Neuper, C. (2010). Combining brain–computer interfaces and assistive technologies: state-of-the-art and challenges. *Frontiers in neuroscience*, 4, 161.

Monge, J. (2022). Higuchi and Katz fractal dimension measures (https://www.mathworks.com/matlabcentral/fileexchange/50290-higuchi-and-katz-fractal-dimension-measures), MATLAB Central File Exchange.

Mora, A., Manyakov, N., Chumerin, N., Van Hulle, M. (2014). Language Model Applications to Spelling with Brain-Computer Interfaces. *Sensors* (Basel, Switzerland). 14. 5967-93. 10.3390/s140405967.

Nicolas, L. F., y Gómez, J. (2012). Brain Computer Interfaces, a Review. Sensors, 12(2), 1211–1279. *MDPI* AG. http://dx.doi.org/10.3390/s120201211

Noble, W. S. (2006). What is a support vector machine? *Nature Biotechnology*, *24*(12), 1565–1567. https://doi.org/10.1038/nbt1206-1565

Polychronaki, G., Ktonas, P., Gatzonis, S., Siatouni, A., Asvestas, P., Tsekou, H., Sakas, D. y Nikita, K. (2010). Comparison of fractal dimension estimation algorithms for epileptic seizure onset detection. *Journal of neural engineering.* 7. 046007. 10.1088/1741-2560/7/4/046007.

Saccá, V., Campolo, M., Mirarchi, D., Gambardella, A., Veltri, P., Morabito, F.C. (2018). On the Classification of EEG Signal by Using an SVM Based Algorithm. In: Esposito, A., Faudez-Zanuy, M., Morabito, F., Pasero, E. (eds) Multidisciplinary Approaches to Neural

Computing. *Smart Innovation, Systems and Technologies*, vol 69. Springer, Cham. https://doi.org/10.1007/978-3-319-56904-8_26

Silvoni, S., Volpato, C., Cavinato, M., Marchetti, M., Priftis, K., Merico, A., Tonin, P., Koutsikos, K., Beverina, F., y Piccione, F. (2009). P300-Based Brain-Computer Interface Communication: Evaluation and Follow-up in Amyotrophic Lateral Sclerosis. *Frontiers in neuroscience*, 3, 60. https://doi.org/10.3389/neuro.20.001.2009

The World Bank. (2019). *Life expectancy at birth, total (years)*. Worldbank.org. https://data.worldbank.org/indicator/SP.DYN.LE00.IN

Van Gerven, M., Farquhar, J., Schaefer, R., Vlek, R., Geuze, J., Nijholt, A., ... y Desain, P. (2009). The brain–computer interface cycle. *Journal of neural engineering*, *6*(4), 041001.

Vareka, L., Mautner, P. (2015). Using the Windowed means paradigm for single trial P300 detection. *38th International Conference on Telecommunications and Signal Processing* (TSP), 2015, pp. 1-4, doi: 10.1109/TSP.2015.7296414.

9. Anexos

A. Clasificador P300 *offline* utilizando el parámetro Varianza

```
clear all;
load p300;
N=240;
Fs=240;
tiempo_win_grande=1;  %en segundos
```

```
tiempo_win_chica=0.3;
win1=Fs*tiempo_win_grande;
win300=Fs*tiempo_win_chica;

Fc1=30;
Fc2=2;

% vetanas 1 s y 300 ms
% CANALES Cz 11 Oz 62  Fz 34
disp('Cz Target')
for i= 1:1:85
   cont=0;
   for j=1:win1:N
   small_signal=Target(i,j:j+win1-1,11);
       for k=1:win300:N-win300
           cont=cont+1;
       cz=small_signal(k:k+win300-1);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y1=filtfilt(b,a,cz);

       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y2=filtfilt(b,a,y1(1,:,1));
       varianza=var(y2);
       var_czT300(i,cont)=varianza;
       end
   end
end

disp('Cz NonTarget')
for i= 1:1:85
   cont=0;
   for j=1:win1:N
   small_signal=NonTarget(i,j:j+win1-1,11);
       for k=1:win300:N-win300
           cont=cont+1;
       cz2=small_signal(k:k+win300-1);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y3=filtfilt(b,a,cz2);

       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y4=filtfilt(b,a,y3(1,:,1));
       varianza=var(y4);
       var_czNT300(i,cont)=varianza;
       end
   end
end

disp('Oz Target')
for i= 1:1:85
   cont=0;
   for j=1:win1:N
   small_signal=Target(i,j:j+win1-1,62);
       for k=1:win300:N-win300
```

```
            cont=cont+1;
        cz3=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y5=filtfilt(b,a,cz3);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y6=filtfilt(b,a,y5(1,:,1));
        varianza=var(y6);
        var_ozT300(i,cont)=varianza;
        end
    end
end
disp('Oz NonTarget')
for i= 1:1:85
    cont=0;
    for j=1:win1:N
    small_signal=NonTarget(i,j:j+win1-1,62);
        for k=1:win300:N-win300
            cont=cont+1;
        cz4=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y7=filtfilt(b,a,cz4);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y8=filtfilt(b,a,y7(1,:,1));
        varianza=var(y8);
        var_ozNT300(i,cont)=varianza;
        end
    end
end

disp('Fz Target')
for i= 1:1:85
    cont=0;
    for j=1:win1:N
    small_signal=Target(i,j:j+win1-1,34);
        for k=1:win300:N-win300
            cont=cont+1;
        cz5=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y9=filtfilt(b,a,cz5);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y10=filtfilt(b,a,y9(1,:,1));
        varianza=var(y10);
        var_fzT300(i,cont)=varianza;
        end
    end
end
disp('Fz NonTarget')
for i= 1:1:85
    cont=0;
    for j=1:win1:N
```

```
    small_signal=NonTarget(i,j:j+win1-1,34);
        for k=1:win300:N-win300
            cont=cont+1;
        cz6=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y11=filtfilt(b,a,cz6);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y12=filtfilt(b,a,y11(1,:,1));
        varianza=var(y12);
        var_fzNT300(i,cont)=varianza;
        end
    end
end

%targets juntos
M=[var_czT300,var_ozT300,var_fzT300;var_czNT300,var_ozNT300,var_fzNT300];

% 30 pruebas
ACCURACY_1=[];
for a=1:1:30
    %LABELS
     labels=[ones((length(M)/2),1);zeros((length(M)/2),1)];

     M_labels=[M,labels];

     %Revolver matriz
     M_labels_mix = M_labels(randperm(size(M_labels, 1)), :);

     %mitad entrenar
     Xtrain=M_labels_mix(1:(length(M)/2),1:end-1);
     %labels mitad entrenar
     Ytrain=M_labels_mix(1:(length(M)/2),end);

     %mitad predecir
     Xtest=M_labels_mix((length(M)/2)+1:end,1:end-1);
     %Y a comparar
     Yreal=M_labels_mix((length(M)/2)+1:end,end);

     %crear y entrenar modelo
     model=fitcsvm(Xtrain,Ytrain);
     %Y a comparar
     Yest=predict(model,Xtest);

     %ACCURACY
      C=confusionmat(Yreal, Yest);
      ACC=(C(1,1)+C(2,2))/length(Yest);
      ACCURACY_1=[ACCURACY_1,ACC];


end

MEANACCURACY=mean(ACCURACY_1);
```

B. Clasificador P300 *offline* utilizando el parámetro Polinomios para Ajuste polinomial de curvas

```
load p300;
N=240;
win=72;
Fs=240;
Fc1=30;
Fc2=2;
g=3;


x = linspace(0,1,72);
disp('Poly Cz Target')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
       cont=cont+1;
   cz=Target(i,j:j+win-1,11);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y1=filtfilt(b,a,cz);

   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y2=filtfilt(b,a,y1(1,:,1));
   p=polyfit(x,y2,g);

   final=start+length(p)-1;
   p_czt(i,start:final)=[p];
   start=final+1;

   end
end


disp('Poly Cz NonTarget')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
  cz2=NonTarget(i,j:j+win-1,11);
  [b,a] = butter(4,[Fc1]/(Fs/2),"low");
  y3=filtfilt(b,a,cz2);
  [b,a] = butter(4,[Fc2]/(Fs/2),"high");
  y4=filtfilt(b,a,y3(1,:,1));
  p=polyfit(x,y4 ,g);
   final=start+length(p)-1;
   p_cznt(i,start:final)=[p];
```

```
   start=final+1;
   end
end

disp('Poly Temporal Target')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
  temp=Target(i,j:j+win-1,34);
  [b,a] = butter(4,[Fc1]/(Fs/2),"low");
  y5=filtfilt(b,a,temp);
  [b,a] = butter(4,[Fc2]/(Fs/2),"high");
  y6=filtfilt(b,a,y5(1,:,1));
  p=polyfit(x,y6,g);
   final=start+length(p)-1;
   p_tempt(i,start:final)=[p];
   start=final+1;
   end
end

disp('Poly Temporal NonTarget')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
  temp2=NonTarget(i,j:j+win-1,34);
  [b,a] = butter(4,[Fc1]/(Fs/2),"low");
  y7=filtfilt(b,a,temp2);
  [b,a] = butter(4,[Fc2]/(Fs/2),"high");
  y8=filtfilt(b,a,y7(1,:,1));
  p= polyfit(x,y8,g);
  final=start+length(p)-1;
  p_tempnt(i,start:final)=[p];
  start=final+1;
   end
end

disp('Poly Iz Target')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
 iz=Target(i,j:j+win-1,62);
 [b,a] = butter(4,[Fc1]/(Fs/2),"low");
 y9=filtfilt(b,a,iz);
 [b,a] = butter(4,[Fc2]/(Fs/2),"high");
 y10=filtfilt(b,a,y9(1,:,1));
 p= polyfit(x,y10,g);
  final=start+length(p)-1;
  p_izt(i,start:final)=[p];
  start=final+1;
   end
```

```
end

disp('Poly Iz NonTarget')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
 iz2=NonTarget(i,j:j+win-1,62);
 [b,a] = butter(4,[Fc1]/(Fs/2),"low");
 y11=filtfilt(b,a,iz2);
 [b,a] = butter(4,[Fc2]/(Fs/2),"high");
 y12=filtfilt(b,a,y11(1,:,1));
 p= polyfit(x,y12,g);
  final=start+length(p)-1;
  p_iznt(i,start:final)=[p];
  start=final+1;
    end
end

M=[p_czt,p_izt,p_izt;p_cznt,p_iznt,p_tempnt];
%%
ACCURACY_1=[];
for a=1:1:30
   %LABELS
    labels=[ones((length(M)/2),1);zeros((length(M)/2),1)];

    M_labels=[M,labels];

    %Revolver matriz
    M_labels_mix = M_labels(randperm(size(M_labels, 1)), :);

    %mitad entrenar
    Xtrain=M_labels_mix(1:(length(M)/2),1:end-1);
    %labels mitad entrenar
    Ytrain=M_labels_mix(1:(length(M)/2),end);

    %mitad predecir
    Xtest=M_labels_mix((length(M)/2)+1:end,1:end-1);
    %Y a comparar
    Yreal=M_labels_mix((length(M)/2)+1:end,end);

    %crear y entrenar modelo
    model=fitcsvm(Xtrain,Ytrain);
    %Y a comparar
    Yest=predict(model,Xtest);

    %ACCURACY
     C=confusionmat(Yreal, Yest);
     ACC=(C(1,1)+C(2,2))/length(Yest);
     %confusionchart(C)
     ACCURACY_1=[ACCURACY_1,ACC];
end
```

```
MEANACCURACY=mean(ACCURACY_1);
```

C. Clasificador P300 *offline* utilizando el parámetro Coherencia

```
load p300;
N=240;
win=72;
Fs=240;
Fc1=30;
Fc2=2;
%returns the magnitude-squared coherence estimate at the frequencies specified
in f.
%cz %oz
%11 62
disp('Coherence Target Window')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
       cont=cont+1;
       cz=Target(i,j:j+win-1,11);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y1=filtfilt(b,a,cz);
       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y2=filtfilt(b,a,y1(1,:,1));

       iz=Target(i,j:j+win-1,62);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y3=filtfilt(b,a,iz);
       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y4=filtfilt(b,a,y3(1,:,1));

       [cxy,f] = mscohere(y2,y4,10,[],90:5:101,Fs);

      final=start+length(cxy)-1;
      coh_t(i,start:final)=[cxy];
      start=final+1;
   end
end

%%
disp('Coherence NonTarget Window')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
       cont=cont+1;
   cz2=NonTarget(i,j:j+win-1,11);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y7=filtfilt(b,a,cz2);
```

```
   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y8=filtfilt(b,a,y7(1,:,1));

   iz2=NonTarget(i,j:j+win-1,62);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y9=filtfilt(b,a,iz2);
   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y10=filtfilt(b,a,y9(1,:,1));
   [cxy2,f2] = mscohere(y8,y10,10,[],90:5:101,Fs);
    final=start+length(cxy)-1;
    coh_nt(i,start:final)=[cxy];
    start=final+1;
   end
%cz iz

end

%%%%plots
promplot=mean(coh_t)
promplot2=mean(coh_nt)
figure,plot(promplot)
hold on, plot (promplot2)
%%
M=[coh_t;coh_nt];
NT=[coh_nt];
T=[coh_t];


ACCURACY_1=[];
for a=1:1:30
   %LABELS
    labels=[ones((length(M)/2),1);zeros((length(M)/2),1)];

    M_labels=[M,labels];

    %Revolver matriz
    M_labels_mix = M_labels(randperm(size(M_labels, 1)), :);

    %mitad entrenar
    Xtrain=M_labels_mix(1:(length(M)/2),1:end-1);
    %labels mitad entrenar
    Ytrain=M_labels_mix(1:(length(M)/2),end);

    %mitad predecir
    Xtest=M_labels_mix((length(M)/2)+1:end,1:end-1);
    %Y a comparar
    Yreal=M_labels_mix((length(M)/2)+1:end,end);

    %crear y entrenar modelo
    model=fitcsvm(Xtrain,Ytrain);
    %Y a comparar
    Yest=predict(model,Xtest);
```

```
    %ACCURACY
     C=confusionmat(Yreal, Yest);
     ACC=(C(1,1)+C(2,2))/length(Yest);
     %confusionchart(C)
     ACCURACY_1=[ACCURACY_1,ACC];
end

MEANACCURACY=mean(ACCURACY_1);
```

D. Clasificador P300 *offline* utilizando el parámetro Correlación

```
load p300;
N=240;
win=72;
Fs=240;
Fc1=30;
Fc2=2;
%returns the magnitude-squared coherence estimate at the frequencies specified
in f.
%cz %oz
%11 62
disp('Coherence Target Window')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
       cont=cont+1;
       cz=Target(i,j:j+win-1,11);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y1=filtfilt(b,a,cz);
       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y2=filtfilt(b,a,y1(1,:,1));

       iz=Target(i,j:j+win-1,62);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y3=filtfilt(b,a,iz);
       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y4=filtfilt(b,a,y3(1,:,1));

       [cxy,f] = mscohere(y2,y4,10,[],90:5:101,Fs);

      final=start+length(cxy)-1;
      coh_t(i,start:final)=[cxy];
      start=final+1;
   end
end

%%
disp('Coherence NonTarget Window')
for i= 1:1:85
   cont=0;
```

```
   start=1;
   for j=1:win:N-win
       cont=cont+1;
   cz2=NonTarget(i,j:j+win-1,11);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y7=filtfilt(b,a,cz2);
   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y8=filtfilt(b,a,y7(1,:,1));

   iz2=NonTarget(i,j:j+win-1,62);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y9=filtfilt(b,a,iz2);
   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y10=filtfilt(b,a,y9(1,:,1));
   [cxy2,f2] = mscohere(y8,y10,10,[],90:5:101,Fs);
    final=start+length(cxy)-1;
    coh_nt(i,start:final)=[cxy];
    start=final+1;
   end
%cz iz

end

%%%%plots
promplot=mean(coh_t)
promplot2=mean(coh_nt)
figure,plot(promplot)
hold on, plot (promplot2)
%%

M=[coh_t;coh_nt];
NT=[coh_nt];
T=[coh_t];


ACCURACY_1=[];
for a=1:1:30
   %LABELS
    labels=[ones((length(M)/2),1);zeros((length(M)/2),1)];

    M_labels=[M,labels];

    %Revolver matriz
    M_labels_mix = M_labels(randperm(size(M_labels, 1)), :);

    %mitad entrenar
    Xtrain=M_labels_mix(1:(length(M)/2),1:end-1);
    %labels mitad entrenar
    Ytrain=M_labels_mix(1:(length(M)/2),end);

    %mitad predecir
    Xtest=M_labels_mix((length(M)/2)+1:end,1:end-1);
    %Y a comparar
```

```
    Yreal=M_labels_mix((length(M)/2)+1:end,end);

    %crear y entrenar modelo
    model=fitcsvm(Xtrain,Ytrain);
    %Y a comparar
    Yest=predict(model,Xtest);

    %ACCURACY
     C=confusionmat(Yreal, Yest);
     ACC=(C(1,1)+C(2,2))/length(Yest);
     %confusionchart(C)
     ACCURACY_1=[ACCURACY_1,ACC];
end

MEANACCURACY=mean(ACCURACY_1);
```

E. Clasificador P300 *offline* utilizando el parámetro Dimensión fractal

```
load p300;
N=240;
win=72;
Fs=240;
Fc1=30;
Fc2=2;

disp('FD Target')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
   cz=Target(i,j:j+win-1,11);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y1=filtfilt(b,a,cz);
   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y2=filtfilt(b,a,y1(1,:,1));
   [xhfd]=hfd(y2,15);


   iz=Target(i,j:j+win-1,62);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y3=filtfilt(b,a,iz);
   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y4=filtfilt(b,a,y3(1,:,1));
   [xhfd2]=hfd(y4,15);

   temp=Target(i,j:j+win-1,34);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y5=filtfilt(b,a,temp);
```

```
   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y6=filtfilt(b,a,y5(1,:,1));
   [xhfd3]=hfd(y6,15);

   fd=[xhfd,xhfd2,xhfd3];
      final=start+size(fd,2)-1;
      fd_t(i,start:final)=[fd];
      start=final+1;
   end
end

%%
disp('FD NonTarget')
for i= 1:1:85
   cont=0;
   start=1;
   for j=1:win:N-win
   cz2=NonTarget(i,j:j+win-1,11);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y7=filtfilt(b,a,cz2);
   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y8=filtfilt(b,a,y7(1,:,1));
   [xhfd4]=hfd(y8,15);


   iz2=NonTarget(i,j:j+win-1,62);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y9=filtfilt(b,a,iz2);
   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y10=filtfilt(b,a,y9(1,:,1));
   [xhfd5]=hfd(y10,15);

   temp2=NonTarget(i,j:j+win-1,34);
   [b,a] = butter(4,[Fc1]/(Fs/2),"low");
   y11=filtfilt(b,a,temp2);
   [b,a] = butter(4,[Fc2]/(Fs/2),"high");
   y12=filtfilt(b,a,y11(1,:,1));
   [xhfd6]=hfd(y12,15);

   fd=[xhfd4,xhfd5,xhfd6];
      final=start+size(fd,2)-1;
      fd_nt(i,start:final)=[fd];
      start=final+1;
   end
end


M=[fd_nt;fd_t];
NT=[fd_nt];
T=[fd_t];


ACCURACY_1=[];
```

```
for a=1:1:30
   %LABELS
    labels=[ones((length(M)/2),1);zeros((length(M)/2),1)];

    M_labels=[M,labels];

    %Revolver matriz
    M_labels_mix = M_labels(randperm(size(M_labels, 1)), :);

    %mitad entrenar
    Xtrain=M_labels_mix(1:(length(M)/2),1:end-1);
    %labels mitad entrenar
    Ytrain=M_labels_mix(1:(length(M)/2),end);

    %mitad predecir
    Xtest=M_labels_mix((length(M)/2)+1:end,1:end-1);
    %Y a comparar
    Yreal=M_labels_mix((length(M)/2)+1:end,end);

    %crear y entrenar modelo
    model=fitcsvm(Xtrain,Ytrain);
    %Y a comparar
    Yest=predict(model,Xtest);

    %ACCURACY
     C=confusionmat(Yreal, Yest);
     ACC=(C(1,1)+C(2,2))/length(Yest);
     %confusionchart(C)
     ACCURACY_1=[ACCURACY_1,ACC];
end

MEANACCURACY=mean(ACCURACY_1);
```

F. Prueba bilateral de medias

```
%Wilcoxon signed rank test
%returns the p-value of a paired, two-sided test for the null hypothesis that x
- y comes from a distribution with zero median.
% accuracy_1 = precisión promedio varianza accuracy_2 = precisión promedio
correlacion accuracy_3 = precisión promedio polinomios accuracy_4 = precisión
promedio dimension fractal accuracy_5 = precisión promedio coherencia

p1 = signrank(ACCURACY_1,ACCURACY_2)
p2 = signrank(ACCURACY_1,ACCURACY_3)
p3 = signrank(ACCURACY_1,ACCURACY_4)
p4 = signrank(ACCURACY_1,ACCURACY_5)
```

G. Clasificador P300 pseudo-*online* utilizando parámetro Varianza

```
run clasificadorvar_windows.m
load Subject_A_Train.mat
N=7794;
win300=80;
win1=240;
Fs=240;
Fc1=30;
Fc2=2;
Signal_one=Signal(1,:,:);
Flash=Flashing(1,:,:);

vant=0;
cont_row=1;

for f=1:1:length(Flash)
    vact=Flash(f);
  if (vact-vant)==1
   small_signal1 = Signal_one(1,f:f+win1-1,11);
   cont_col = 1;
       for j=1:win300:win1-win300+1
           one3=small_signal1(j:j+win300-1);
           onedouble=double(one3);
           [b,a] = butter(4,[Fc1]/(Fs/2),"low");
           y1=filtfilt(b,a,onedouble);
           [b,a] = butter(4,[Fc2]/(Fs/2),"high");
           y2=filtfilt(b,a,y1(1,:,1));
           varianza3=var(y2);
           var_cz300(cont_row,cont_col)=varianza3;
           cont_col = cont_col +1;
       end
       cont_row= cont_row+1;
  end
  vant=vact;
end

cont_row=1;

for f=1:1:length(Flash)
    vact=Flash(f);
  if (vact-vant)==1
   small_signal2 = Signal_one(1,f:f+win1-1,34);
   cont_col = 1;
       for j=1:win300:win1-win300+1
           one3=small_signal2(j:j+win300-1);
           onedouble=double(one3);
           [b,a] = butter(4,[Fc1]/(Fs/2),"low");
           y1=filtfilt(b,a,onedouble);
           [b,a] = butter(4,[Fc2]/(Fs/2),"high");
           y2=filtfilt(b,a,y1(1,:,1));
```

```
            varianza3=var(y2);
            var_fz300(cont_row,cont_col)=varianza3;
            cont_col = cont_col +1;
        end
        cont_row= cont_row+1;
  end
  vant=vact;
end

cont_row=1;

for f=1:1:length(Flash)
    vact=Flash(f);
  if (vact-vant)==1
   small_signal3 = Signal_one(1,f:f+win1-1,62);
   cont_col = 1;
        for j=1:win300:win1-win300+1
            one3=small_signal3(j:j+win300-1);
            onedouble=double(one3);
            [b,a] = butter(4,[Fc1]/(Fs/2),"low");
            y1=filtfilt(b,a,onedouble);
            [b,a] = butter(4,[Fc2]/(Fs/2),"high");
            y2=filtfilt(b,a,y1(1,:,1));
            varianza3=var(y2);
            var_oz300(cont_row,cont_col)=varianza3;
            cont_col = cont_col +1;
        end
        cont_row= cont_row+1;
  end
  vant=vact;
end

F_VAR=[var_cz300,var_oz300,var_fz300];

%clasificador
Yest2=predict(model,F_VAR);

   for e=1:1:length(F_VAR)
        small_stim = StimulusType(1,e:e+win1-1);
        for j=1:win300:win1-win300+1
            one=small_stim(j:j+win300-1);
            suma=sum(one);
            STIMULUS(cont)=suma;
        end
        cont= cont+1;
   end
pruebasuma=find(STIMULUS>0);
pruebasuma2=zeros(1,length(F_VAR));

pruebasuma2(pruebasuma)=1;
Yreal2=pruebasuma2';

C=confusionmat(Yest2, Yreal2);
```

```
ACCURACY_PSEUDO=(C(1,1)+C(2,2))/length(Yest2);
```

H. Programa para adquisición de interfaz

```
% Conexión con BioRadio
clear all, close all
asmInfo =
NET.addAssembly('C:\Users\yanis\OneDrive\Documentos\MATLAB\p300\BIORADIO\BioRadi
oSDK.dll')  %Dirección del achivo BioRadioSDK
deviceManager = GLNeuroTech.Devices.BioRadio.BioRadioDeviceManager;
macID = int64(hex2dec('ECFE7E12AB5E'));  % %macID
myDevice = deviceManager.GetBluetoothDevice(macID);
Fs=double(myDevice.BioPotentialSignals.SamplesPerSecond);   %Fs
numEnabledBPChannels=double(myDevice.BioPotentialSignals.Count);
%Número de canales

%% 1 R 3 V 5 A
Fs=500;
imgs = cell(5,1);

imgs{1} = imread('rojo2.png');
imgs{2} = imread('grismatlab.png');
imgs{3} = imread('verde2.png');
imgs{4} = imread('gris.png');
imgs{5} = imread('azul2.png');

figure;
axis ij
pause(2)
%myDevice.StartAcquisition;
prueba=[1 1 1 3 3 3 5 5 5];
tic
for a=1:20
    %mostrar cuadros
    subplot(1,5,1), imshow(imgs{4});
    subplot(1,5,2), imshow(imgs{2});
    subplot(1,5,3), imshow(imgs{4});
    subplot(1,5,4), imshow(imgs{2});
    subplot(1,5,5), imshow(imgs{4});

    i=randsample(prueba,1);
    index_color(a)=i; %guardar color
    pause(.3+rand*.5) %tiempo cambio 0.3 0.8
    subplot(1,5,i)
    imshow(imgs{i});
   % toc
    stim_time=toc;
    pause(.5) %tiempo cuadro
    b=round(stim_time*Fs);
    index_time(b)=1; %guardar tiempo
    in_t=find(index_time==1);
```

```
end
    subplot(1,5,1), imshow(imgs{4});
    subplot(1,5,2), imshow(imgs{2});
    subplot(1,5,3), imshow(imgs{4});
    subplot(1,5,4), imshow(imgs{2});
    subplot(1,5,5), imshow(imgs{4});

pause(2)

%myDevice.StopAcquisition;

for c=0:2
BioPotentialSignals(:,c+1) =
myDevice.BioPotentialSignals.Item(c).GetScaledValueArray.double';  %empieza en
canal 0 %% for para que haga los canales
end

if length(BioPotentialSignals)<length(index_time)
    errordlg("Error en la medicion")
else
    msgbox("Medición realizada con éxito")
    save("s4_120922_a2.mat","index_color","index_time","BioPotentialSignals")
end
% num sujeto, fecha, color sesiones de prueba s1_290822_r
clear BioPotentialSignals index_color index_time
close all

%% Desconexión del dispositivo
myDevice.Disconnect
```

I. Programa de entrenamiento *offline* de interfaz

```
clear all, %close all
load s4_120922_v2.mat

%   tres
rojo=1;
verde=3;
azul=5;
%% DIVISION DE SEÑALES POR COLORES EN TODOS LOS CANALES
in_t=find(index_time==1);

color=rojo;
t=0;
nt=0;
target=[];
non_target=[];
for i=1:length(in_t)
    if index_color(i)==color
        t=t+1;
```

```
        target_rojo(:,:,t)=BioPotentialSignals;
    else
        nt=nt+1;
        non_target_rojo(:,:,nt)=BioPotentialSignals;
    end
end

color=verde;
t=0;
nt=0;
target=[];
non_target=[];
for i=1:length(in_t)
    if index_color(i)==color
        t=t+1;
        target_verde(:,:,t)=BioPotentialSignals;
    else
        nt=nt+1;
        non_target_verde(:,:,nt)=BioPotentialSignals;
    end
end

color=azul;
t=0;
nt=0;
target=[];
non_target=[];
for i=1:length(in_t)
    if index_color(i)==color
        t=t+1;
        target_azul(:,:,t)=BioPotentialSignals;
    else
        nt=nt+1;
        non_target_azul(:,:,nt)=BioPotentialSignals;
    end
end
%% PERMUTE
target_rojo2 = permute(target_rojo,[3 1 2]);
non_target_rojo2 = permute(non_target_rojo,[3 1 2]);
target_verde2 = permute(target_verde,[3 1 2]);
non_target_verde2 = permute(non_target_verde,[3 1 2]);
target_azul2 = permute(target_azul,[3 1 2]);
non_target_azul2 = permute(non_target_azul,[3 1 2]);
%% PROCESAMIENTO
N=500;
Fs=500;
tiempo_win_grande=1;  %en segundos
tiempo_win_chica=0.3;
win1=Fs*tiempo_win_grande;
win300=Fs*tiempo_win_chica;

Fc1=30;
Fc2=2;
```

```
% ROJO

disp('Rojo Target CH 1')
for i= 1:1:size(target_rojo2,1)
   cont=0;
   for j=1:win1:N
   small_signal=target_rojo2(i,j:j+win1-1,1);
       for k=1:win300:N-win300
           cont=cont+1;
       cz=small_signal(k:k+win300-1);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y1=filtfilt(b,a,cz);

       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y2=filtfilt(b,a,y1(1,:,1));
       varianza=var(y2);
       var_rojot1(i,cont)=varianza;
       end
   end
end

disp('Rojo Target CH 2')
for i= 1:1:size(target_rojo2,1)
   cont=0;
   for j=1:win1:N
   small_signal=target_rojo2(i,j:j+win1-1,2);
       for k=1:win300:N-win300
           cont=cont+1;
       cz=small_signal(k:k+win300-1);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y1=filtfilt(b,a,cz);

       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y2=filtfilt(b,a,y1(1,:,1));
       varianza=var(y2);
       var_rojot2(i,cont)=varianza;
       end
   end
end

disp('Rojo Target CH 3')
for i= 1:1:size(target_rojo2,1)
   cont=0;
   for j=1:win1:N
   small_signal=target_rojo2(i,j:j+win1-1,3);
       for k=1:win300:N-win300
           cont=cont+1;
       cz=small_signal(k:k+win300-1);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y1=filtfilt(b,a,cz);

       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y2=filtfilt(b,a,y1(1,:,1));
```

```
        varianza=var(y2);
        var_rojot3(i,cont)=varianza;
        end
    end
end
%%
disp('Rojo NonTarget CH 1')
for i= 1:1:size(non_target_rojo2,1)
    cont=0;
    for j=1:win1:N
    small_signal=non_target_rojo2(i,j:j+win1-1,1);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_rojont1(i,cont)=varianza;
        end
    end
end
disp('Rojo NonTarget CH 2')
for i= 1:1:size(non_target_rojo2,1)
    cont=0;
    for j=1:win1:N
    small_signal=non_target_rojo2(i,j:j+win1-1,2);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_rojont2(i,cont)=varianza;
        end
    end
end

disp('Rojo NonTarget CH 3')
for i= 1:1:size(non_target_rojo2,1)
    cont=0;
    for j=1:win1:N
    small_signal=non_target_rojo2(i,j:j+win1-1,3);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);
```

```
        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_rojont3(i,cont)=varianza;
        end
    end
end
%% VERDE
disp('Verde Target CH 1')
for i= 1:1:size(target_verde2,1)
    cont=0;
    for j=1:win1:N
    small_signal=target_verde2(i,j:j+win1-1,1);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_verdet1(i,cont)=varianza;
        end
    end
end

disp('Verde Target CH 2')
for i= 1:1:size(target_verde2,1)
    cont=0;
    for j=1:win1:N
    small_signal=target_verde2(i,j:j+win1-1,2);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_verdet2(i,cont)=varianza;
        end
    end
end

disp('Verde Target CH 3')
for i= 1:1:size(target_verde2,1)
    cont=0;
    for j=1:win1:N
    small_signal=target_verde2(i,j:j+win1-1,3);
        for k=1:win300:N-win300
```

```
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_verdet3(i,cont)=varianza;
        end
    end
end
%%
disp('Verde NonTarget CH 1')
for i= 1:1:size(non_target_verde2,1)
    cont=0;
    for j=1:win1:N
    small_signal=non_target_verde2(i,j:j+win1-1,1);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_verdent1(i,cont)=varianza;
        end
    end
end
disp('Verde NonTarget CH 2')
for i= 1:1:size(non_target_verde2,1)
    cont=0;
    for j=1:win1:N
    small_signal=non_target_verde2(i,j:j+win1-1,2);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_verdent2(i,cont)=varianza;
        end
    end
end

disp('Verde NonTarget CH 3')
for i= 1:1:size(non_target_verde2,1)
    cont=0;
```

```
    for j=1:win1:N
    small_signal=non_target_verde2(i,j:j+win1-1,3);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_verdent3(i,cont)=varianza;
        end
    end
end

%% AZUL

disp('Azul Target CH 1')
for i= 1:1:size(target_azul2,1)
    cont=0;
    for j=1:win1:N
    small_signal=target_azul2(i,j:j+win1-1,1);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_azult1(i,cont)=varianza;
        end
    end
end

disp('Azul Target CH 2')
for i= 1:1:size(target_azul2,1)
    cont=0;
    for j=1:win1:N
    small_signal=target_azul2(i,j:j+win1-1,2);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_azult2(i,cont)=varianza;
        end
```

```
   end
end

disp('Azul Target CH 3')
for i= 1:1:size(target_azul2,1)
   cont=0;
   for j=1:win1:N
   small_signal=target_azul2(i,j:j+win1-1,3);
       for k=1:win300:N-win300
           cont=cont+1;
       cz=small_signal(k:k+win300-1);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y1=filtfilt(b,a,cz);

       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y2=filtfilt(b,a,y1(1,:,1));
       varianza=var(y2);
       var_azult3(i,cont)=varianza;
       end
   end
end
%%
disp('Azul NonTarget CH 1')
for i= 1:1:size(non_target_azul2,1)
   cont=0;
   for j=1:win1:N
   small_signal=non_target_azul2(i,j:j+win1-1,1);
       for k=1:win300:N-win300
           cont=cont+1;
       cz=small_signal(k:k+win300-1);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y1=filtfilt(b,a,cz);

       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y2=filtfilt(b,a,y1(1,:,1));
       varianza=var(y2);
       var_azulnt1(i,cont)=varianza;
       end
   end
end
disp('Azul NonTarget CH 2')
for i= 1:1:size(non_target_azul2,1)
   cont=0;
   for j=1:win1:N
   small_signal=non_target_azul2(i,j:j+win1-1,2);
       for k=1:win300:N-win300
           cont=cont+1;
       cz=small_signal(k:k+win300-1);
       [b,a] = butter(4,[Fc1]/(Fs/2),"low");
       y1=filtfilt(b,a,cz);

       [b,a] = butter(4,[Fc2]/(Fs/2),"high");
       y2=filtfilt(b,a,y1(1,:,1));
```

```
        varianza=var(y2);
        var_azulnt2(i,cont)=varianza;
        end
    end
end

disp('Azul NonTarget CH 3')
for i= 1:1:size(non_target_azul2,1)
    cont=0;
    for j=1:win1:N
    small_signal=non_target_azul2(i,j:j+win1-1,3);
        for k=1:win300:N-win300
            cont=cont+1;
        cz=small_signal(k:k+win300-1);
        [b,a] = butter(4,[Fc1]/(Fs/2),"low");
        y1=filtfilt(b,a,cz);

        [b,a] = butter(4,[Fc2]/(Fs/2),"high");
        y2=filtfilt(b,a,y1(1,:,1));
        varianza=var(y2);
        var_azulnt3(i,cont)=varianza;
        end
    end
end
%%

M=[var_rojot1, var_rojot2, var_rojot3;var_rojont1, var_rojont2,
var_rojont3;var_verdet1,var_verdet2,var_verdet3;var_verdent1,var_verdent2,var_ve
rdent3;var_azult1,var_azult2,var_azult3;var_azulnt1,var_azulnt2,var_azulnt3];


ACCURACY_1=[];
for a=1:1:30
    %LABELS
     labels=[ones((length(M)/2),1);zeros((length(M)/2),1)];

     M_labels=[M,labels];

     %Revolver matriz
     M_labels_mix = M_labels(randperm(size(M_labels, 1)), :);

     %mitad entrenar
     Xtrain=M_labels_mix(1:(length(M)/2),1:end-1);
     %labels mitad entrenar
     Ytrain=M_labels_mix(1:(length(M)/2),end);

     %mitad predecir
     Xtest=M_labels_mix((length(M)/2)+1:end,1:end-1);
     %Y a comparar
     Yreal=M_labels_mix((length(M)/2)+1:end,end);

     %crear y entrenar modelo
     model=fitcsvm(Xtrain,Ytrain);
```

```
    %Y a comparar
    Yest=predict(model,Xtest);

    %ACCURACY
     C=confusionmat(Yreal, Yest);
     ACC=(C(1,1)+C(2,2))/length(Yest);
     ACCURACY_1=[ACCURACY_1,ACC];
end

MEANACCURACY=mean(ACCURACY_1);
```

J. Programa ejemplo para conexión de clasificador con Arduino

Sección de código que se puede agregar a un clasificador donde Yest es el resultado estimado.

```
%% arduino
%correr una vez esta linea y borrar todo cada que se crea una nueva variable ar
arduinosetup()
%%
ar=arduino()
for b=1:1:length(Yest)
    switch Yest(b)
        case 1
            writeDigitalPin(ar, 'D13', 1);
              pause(0.5);
            disp('positive one')

        case 0
            writeDigitalPin(ar, 'D13', 0);
              pause(0.5);
            disp('zero')
    end
end
```