DESARROLLO DE ALGORITMO DE RECONOCIMIENTO DE MOVIMIENTOS DE LA MANO BASADO EN APRENDIZAJE AUTOMÁTICO

LUISA FERNANDA BEJARANO MIRANDA

UNIVERSIDAD SANTO TOMÁS
FACULTAD DE INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.
2021

DESARROLLO DE ALGORITMO DE RECONOCIMIENTO DE MOVIMIENTOS DE LA MANO BASADO EN APRENDIZAJE AUTOMÁTICO

LUISA FERNANDA BEJARANO MIRANDA

Proyecto de grado presentado como requisito para optar al título de
INGENIERO ELECTRÓNICO

Director: Ing. Juan Manuel Calderón Chávez

UNIVERSIDAD SANTO TOMÁS DE AQUINO
FACULTAD DE INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.
2021

Dedicado a todas aquellas personas que hicieron parte activa de este proyecto.

AGRADECIMIENTOS

Agradezco el desarrollo de este proyecto principalmente a mi director Juan Manuel Calderón Chávez, ya que sin su apoyo y acompañamiento constante no hubiera sido posible llevarlo a cabo. Agradezco también a mi familia por su constante ánimo y a los docentes que en su momento me ayudaron como lo fue Oscar Mauricio Gelves Lizarazo y Jairo Alejandro Rodríguez Martínez.

CONTENIDO

RESUMEN

Con el propósito de ahondar en el campo de la inteligencia artificial aplicado a sistemas bio-inspirados, se desarrolló un algoritmo que es capaz de reconocer los movimientos de la mano usando una red neuronal artificial y un sensor Myo.

Puesto que existen muchas soluciones a nivel de hardware para las personas en condición de discapacidad de miembros superiores, se piensa en la posibilidad de desarrollar un algoritmo que permita el reconocimiento y clasificación de los movimientos realizados.

Este proyecto propone el desarrollo de un algoritmo de aprendizaje automático que permite clasificar las señales recibidas por una serie de electrodos de superficie para identificar el movimiento que ha realizado el usuario.

Este documento detalla la investigación realizada acerca de la generación de las señales electromiográficas, luego se explica el proceso de configuración y muestreo, usando el sensor Myo, para crear un dataset que posteriormente será usado para clasificar las señales. A continuación, se explica el funcionamiento del algoritmo, el proceso de adecuación de las señales, filtrado y entrenamiento de la red neuronal artificial. Se presenta el desarrollo y funcionamiento de la interfaz gráfica diseñada para facilitar la interacción del usuario con el algoritmo. También, se agrega un algoritmo estadístico de prueba con el fin comparar los resultados obtenidos de la red neuronal con otras técnicas de clasificación, y así, definir si el sistema mejora la identificación de los movimientos. Finalmente, se realizan observaciones de los resultados y comentarios para nuevos desarrollos.

Dado que las redes neuronales son un sistema de clasificación más robusto que los algoritmos estadísticos, se espera que los resultados de la red neuronal indiquen un porcentaje de acierto significativamente superior a los resultados obtenidos por el algoritmo estadístico. Para determinar este factor, se analizarán los resultados al ejecutar los algoritmos de forma simultánea para identificar 100 muestras de cada movimiento.

Los resultados del proyecto indicaron un porcentaje de acierto del 66.28% para la red neuronal diseñada, mientras el algoritmo estadístico obtuvo tan solo el 31.42% de aciertos. Lo que lleva a la conclusión de que un algoritmo de aprendizaje automático es una buena opción para identificar las señales de un usuario con discapacidad. Esto abre la puerta a la posibilidad de implementar este tipo de algoritmos sobre el control de prótesis de brazo y así promover la independencia de las personas con discapacidad de miembros superiores.

INTRODUCCIÓN

Conseguir el desarrollo de mecanismos que permitan al ser humano realizar actividades cotidianas a pesar de tener alguna discapacidad es uno de los objetivos más plausibles de la biónica. En este campo se han llevado a cabo muchos proyectos de diseño de prótesis con altos grados de libertad para que la experiencia de los usuarios sea lo más cómoda posible, también se han diseñado algoritmos para el control de dichas prótesis.

Los proyectos enfocados a rehabilitación implican un gran trabajo de investigación, principalmente aquellos que están vinculados al estudio de la generación de las señales musculares, conocidas como señales electromiográficas (EMG), y su relación con los movimientos del cuerpo. Puesto que para cada movimiento que realiza el ser humano hay diferentes músculos asociados, es importante analizar la función de cada uno de ellos y las señales que emite al generarse un impulso voluntario.

Las señales EMG varían entre una persona y otra dependiendo de sus características físicas e incluso de su edad, por esta razón no es posible generalizar por completo las señales del cuerpo humano. Cuando una persona presenta discapacidad, sus músculos no emiten las mismas señales que emite una persona sana, sus músculos envían de forma diferente los impulsos eléctricos [1]. Dado que las soluciones tecnológicas para una persona discapacitada son personalizadas en alto grado, es importante que los desarrollos sean fácilmente adaptables. Esto significa un reto importante ya que no se trata de una máquina que realice acciones repetitivas, sino que debe adaptarse a cada situación.

Uno de los avances tecnológicos con mayor auge los últimos años es la inteligencia artificial, esta ha demostrado ser muy versátil y tener grandes capacidades para procesar información. El cerebro humano, bien conocido como la “maquina perfecta” [2], tiene una forma particular de relacionar información y tomar decisiones. No se trata de una sola unidad de procesamiento, sino de muchas unidades independientes que se comunican entre sí para resolver problemas complejos. Este concepto ha sido la base para el desarrollo de algoritmos que, usando unidades independientes conectadas entre sí, pueden ajustarse para aprender a reconocer patrones.

La adaptabilidad de las redes neuronales la convierten en un candidato ideal para solucionar el problema de identificación de señales musculares en personas con discapacidad ya que, mediante el entrenamiento, es posible enseñarle al algoritmo a reconocer patrones en los impulsos eléctricos de un individuo sin necesidad de compararlo con otro ni cambiar la lógica del sistema de identificación para obtener resultados precisos.

Este trabajo de investigación busca mostrar de forma práctica los resultados que se pueden obtener al procesar señales musculares usando redes neuronales y las posibilidades que existen al implementar algoritmos adaptables para solucionar problemas complejos. También ofrece un aporte al área de inteligencia artificial, en cuanto a la aplicación de algoritmos de aprendizaje automático, que forma parte de los grupos MEM y GED pertenecientes a la Facultad de Ingeniería Electrónica de la Universidad Santo Tomás.

1. PROBLEMA

En Colombia gran cantidad de personas pierden miembros de su cuerpo debido a accidentes, enfermedades o en combate. Según un estudio realizado para la Revista Colombiana de Cirugía [3] cerca del 26% de los heridos en combate que ingresaron en el hospital en el periodo transcurrido entre el 1º de enero de 2012 y el 31 de julio de 2014 necesitaron amputación de sus miembros. Se realizaron un total de 251 amputaciones únicamente en el Hospital Militar Central de Bogotá. Por otro lado, según un estudio realizado en hospitales de EEUU, 1 de cada 190 estadounidenses vive sin alguna de sus extremidades y se proyecta que para el año 2050 la cantidad de personas en este estado aumentarán a 3.6 millones [4]. Para el año 2005 cerca de 1,6 millones de personas vivían sin alguna de sus extremidades donde al menos el 38% había perdido sus miembros debido a alguna enfermedad cardiovascular [4].

Las cifras mencionadas ofrecen un indicio de la necesidad de proponer soluciones que permitan mejorar la calidad de vida de las personas en condición de discapacidad. El uso de prótesis ha sido uno de los métodos más utilizados por el sistema de salud para ayudar a las personas que han perdido miembros superiores. No obstante, gran cantidad de prótesis usadas en Colombia son rígidas y ofrecen poca libertad de movimiento al usuario, lo que dificulta la realización de muchas actividades cotidianas.

Diversas empresas, tales como Bioparx y Prótesis Avanzadas, se han dedicado a realizar prótesis que permitan al usuario realizar una variedad de movimientos. Para construirlas, utilizan sensores electromiográficos y algoritmos para el procesamiento de las señales recibidas. A la hora de desarrollar los algoritmos, una de las mayores dificultades radica en conseguir que sea capaz de identificar de manera adecuada el movimiento que indica la contracción muscular. Sin embargo, cada persona emite señales ligeramente diferentes al realizar el mismo movimiento, por lo cual es difícil asegurar que un algoritmo sea capaz de adaptarse a cualquier usuario.

Teniendo presentes estos aspectos, la pregunta de investigación para este proyecto es ***¿qué algoritmo puede realizar el procesamiento de señales mioeléctricas y, en caso de ser necesario, reajustarse para que el error en los movimientos indicados por el usuario disminuya?***

2. ANTECEDENTES

Los estudios realizados en el área de la biología han impulsado importantes avances en el área de la biónica, un claro ejemplo es que conocer el proceso que efectúa el cuerpo humano para ejecutar un movimiento muscular ha llevado a materializar ideas que buscan solucionar las principales dificultades de las personas que tienen alguna discapacidad física. La importante relación entre estas dos áreas se ve expuesta a continuación, donde se expone el comportamiento muscular tras un impulso eléctrico y algunos proyectos que han tenido como base estos conceptos.

Las señales EMG permiten registrar los potenciales eléctricos emitidos por los músculos estriados como respuesta a un impulso eléctrico. Los músculos estriados son aquellos que están compuestos por filamentos conocidos como fibras musculares estriadas, dichas fibras pueden llegar a medir hasta 30 cm de largo con un diámetro menor a 100 µm. Cada una de las fibras recibe señales eléctricas mediante unos conectores conocidos como neurofibrillas. El tiempo en el que ocurre la contracción muscular después de recibir la señal eléctrica es de 5 a 10 ms para las llamadas “fibras rápidas” y de unos 100 ms para las llamadas “fibras lentas”. El potencial con el cual se contraen las fibras musculares depende del balance de iones de la membrana y es de unos 80 mV [5].
Una unidad motora está compuesta por un grupo de placas motoras. Dicha unidad recibe las señales mediante un axón, cada axón tiene las neurofibrillas de la unidad motora (Figura 1). Esta distribución permite activar de manera asincrónica las neurofibrillas, por lo tanto, la unidad completa tarda en contraerse un poco más de lo que tardaría una sola fibra [5].

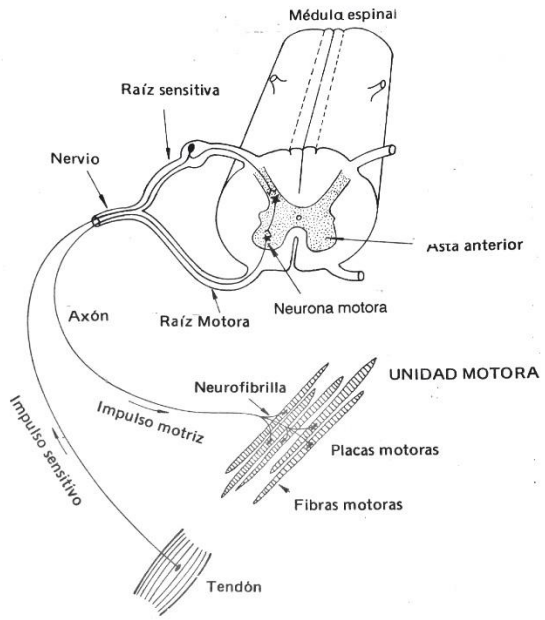

Figura 1. Inervación del musculo: Unidad Motora. Tomado de: C. D. Aguila, Electromedicina, 2 ed., Buenos Aires: Presencia Ltda, 1994, pp. 225-238.

Los impulsos eléctricos son enviados desde el sistema nervioso central y las señales son transmitidas por “motoneuronas” que se encuentran ubicadas en la médula espinal. Cuando el impulso eléctrico llega a las neurofibrillas, estas liberan acetilcolina sobre un receptor químico que tiene cada una de las fibras musculares. Esta emisión hace que la fibra empiece a contraerse.

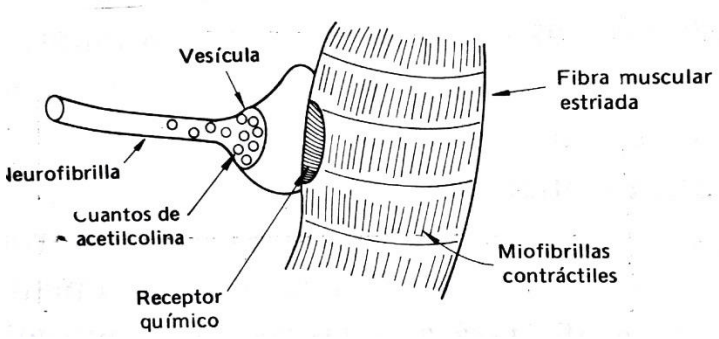

Figura 2. Conexión de una neurofibrilla a una fibra muscular. Tomado de: C. D. Aguila, Electromedicina, 2 ed., Buenos Aires: Presencia Ltda, 1994, pp. 225-238.

En la figura 3 se puede observar el comportamiento de un anca de rana cuando se le aplica un impulso eléctrico. Tras el impulso, se presenta un tiempo de retardo de aproximadamente 4ms antes de que el musculo inicie la contracción, y tarda cerca de 20ms en contraerse completamente. Esto indica que los músculos no reaccionan de forma instantánea al aplicarse un impulso eléctrico sino que presenta un tiempo

de retardo relacionado con el tiempo de transmisión de las motoneuronas y el tiempo de liberación de la acetilcolina sobre las fibras musculares.

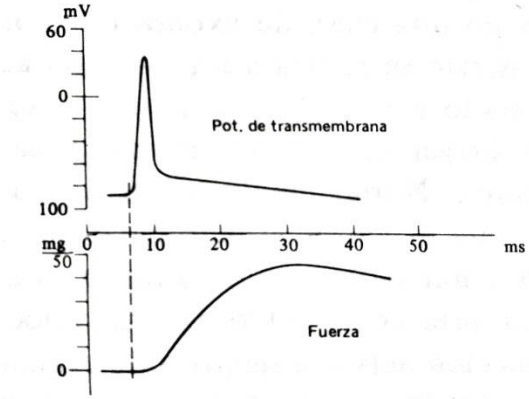

Figura 3. Potencial de la contracción muscular de un anca de rana. Tomado de: C. D. Aguila, Electromedicina, 2 ed., Buenos Aires: Presencia Ltda, 1994, pp. 225-238.

Los impulsos eléctricos transmitidos desde la motoneurona tienen una velocidad de 0,2 a 120 m/s. También hay que tener presente que cada tipo de neurona tiene una velocidad diferente dependiendo de su función. En la tabla 1 podemos observar que la velocidad de conducción para una fibra nerviosa motora es de 70 a 120 m/s y su potencial de acción dura aproximadamente de 0,4 a 0,5 ms.

TIPO DE FIBRA	DIÁMETRO µ	VELOCIDAD DE CONDUCCIÓN M/SEG	DURACIÓN DE LA ESPIGA DEL POTENCIAL DE ACCIÓN MSEG	FUNCIÓN
Aα	13-22	70-120	0,4-0,5	Motora, Propioceptores musculares
Aβ	8-13	40-70	0,4-0,6	Tacto, Presión, Cinestesia
Aγ	4-8	15-40	0,5-0,7	Tacto, excitación motora de los husos musculares
Aδ	1-4	5-15	0,6-1	Dolor, Calor, Frio, Presión
B	1-3	3-14	1,2	Sistema Vegetativo

C	0,2-1	0,2-2	2	Dolor, Prurito, Sistema Vegetativo

Tabla 1. Clasificación de las fibras nerviosas. Tomado de: C. D. Aguila, Electromedicina, 2 ed., Buenos Aires: Presencia Ltda, 1994, pp. 225-238.

Una de las aplicaciones más conocidas del procesamiento de las señales electromiográficas es el control de prótesis para personas amputadas. En la figura 4 de puede contemplar el diseño de un sistema que cuenta con dos etapas, la primera es una etapa de clasificación de señales en tiempo real que cuenta con 4 subniveles: 1) adecuación de las señales, 2) clasificación de características, 3) red neuronal artificial y 4) control de actuadores. Esta primera etapa es la encargada de realizar la clasificación en tiempo real para el control de la prótesis de brazo. La segunda etapa es de entrenamiento y cuenta con 3 subniveles: 1) base de datos de patrones, 2) algoritmo de entrenamiento y 3) base de datos de pesos. Esta etapa se encarga de preparar la red neuronal artificial para que reconocer los movimientos realizados por el usuario. Las dos etapas principales se relacionan al compartir información de las bases de datos de pesos de la red neuronal y los patrones de movimientos realizados [6].

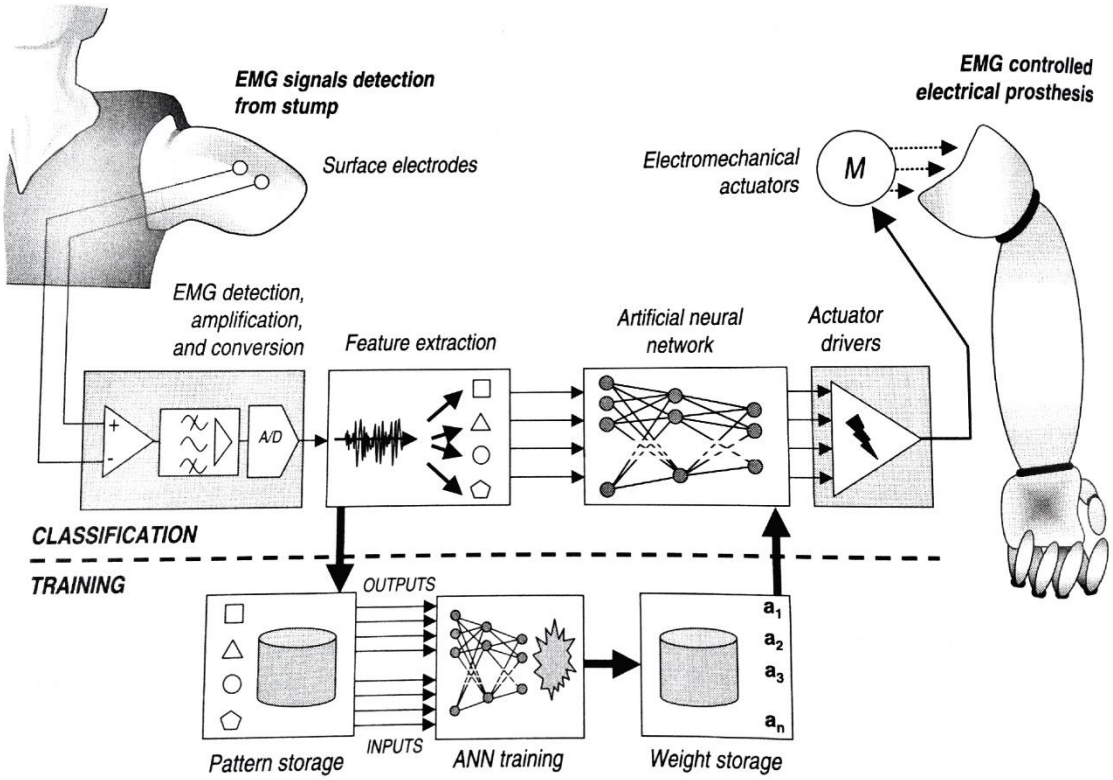

Figura 4. Diagrama de bloques de un sistema de control para prótesis de brazo usando redes neuronales artificiales. Tomado de: J. Moore y G. Zouridakis, Biomedical Technology and Devices Handbook, United States of America: CRC Press, 2004.

Hablando de proyectos basados en lectura de señales para miembros superiores se encuentra el proyecto realizado por G. Feleke y C. Guan para la colaboración humano-máquina, aquí se realiza un proceso para la predicción de los movimientos del brazo de un operario. En este proyecto se realizó la toma de datos usando electrodos superficiales y para procesar las señales se usó un algoritmo de regresión lineal [7].

Este proyecto consiguió una buena identificación de las señales EMG, sin embargo, la exactitud puede verse afectada al utilizar como método de procesamiento de señales la regresión lineal ya que, las señales de cada persona difieren en algunos parámetros por lo cual es difícil garantizar que el movimiento sea predicho correctamente si se cambia de operario, como se ha mencionado previamente.

Otros proyectos se han enfocado en el estudio de las señales obtenidas tras diferentes movimientos de la mano teniendo en cuenta la posición del brazo. Para el proyecto de M. Jochumsen, A. Waris y E. Kamavuako, respecto al efecto de la posición del brazo en la clasificación de señales musculares, se realizaron pruebas en 8 personas sanas con sensores intramusculares para conocer la diferencia de las señales producidas por el brazo en diferentes posiciones, este estudio arrojó que es más fácil determinar el tipo de movimiento que realiza la mano si se conoce la posición del brazo. Existe un punto “neutro” donde las señales producidas debido a la fuerza realizada por el antebrazo son mínimas, dicha posición es con el brazo suspendido de forma vertical, lo que indica que es la mejor forma de llevar a cabo el proceso de adquisición de datos. Sin embargo, el estudio señaló que las pruebas se realizaron únicamente en personas sanas y para poder diseñar una prótesis es importante tener en cuenta las señales de una persona discapacitada ya que este tipo de trabajos están dirigidos a dicha población [8].

Aunque lo más común es diseñar prótesis usando como referencia las señales mioeléctricas obtenidas directamente del antebrazo, existen gran cantidad de pruebas que se han realizado para encontrar diferentes formas de controlar los mecanismos; una de ellas es a través de señales EEG, es decir, mediante señales neuronales (electroencefalográficas). Esta es otra técnica que ha mostrado buenos resultados para el control de prótesis, el proyecto de Emiliano Noce y Alberto Dellacasa titulado *EMG and ENG-envelope pattern recognition for prosthetic hand control,* publicado en 2019 en Journal of Neuroscience Methods, indica que el sistema de control de prótesis basado en señales EEG tiene una precisión del 70% en la predicción de los movimientos [9]. Para este proyecto se utilizaron algoritmos similares a los usados para el procesamiento de señales EMG. Sin embargo, el uso de algoritmos de aprendizaje automático puede llevar a un avance en el campo de la inteligencia artificial.

Para la adquisición, procesamiento y análisis de la información obtenida por los sensores mioeléctricos existe gran cantidad de algoritmos, entre los cuales se encuentran la red neuronal tipo perceptrón multicapa, la regresión lineal, la máquina de soporte vectorial, entre otros, y cada uno ofrece diferentes ventajas.

Según un estudio realizado en 2017 por Emiliano Dellacasa y Emanuelle Gruppioni titulado *NLR, MLP, SVM, and LDA: a comparative analysis on EMG data from people with trans-radial amputation* y publicado en la revista *Journal of*

NeuroEngineering and Rehabilitation se presentaron los resultados de un análisis comparativo de diferentes algoritmos de procesamiento de señales EMG en personas con amputaciones transradiales. Se realizaron las pruebas con 30 personas que realizaban 5 gestos con las manos. Los algoritmos se probaron a través de un análisis estadístico basado en la prueba Wilcoxon Signed-Rank y el consumo de recursos computacionales durante la ejecución. El análisis arrojó como resultado que el algoritmo de mejor rendimiento es la maquina de soporte vectorial seguida por el Perceptrón Multicapa (ver la Tabla 2) [10].

Algoritmo de clasificación	F1Score	Número de parámetros de clasificación	EOF
NLR (100Hz)	92,0 (6,1s)	362 (41s)	95,5 (3,4s)
MLP (100Hz)	92,5 (5,9s)	1654 (605s)	94,8 (3,2s)
SVM (10Hz)	89,5 (7,3s)	1361 (648s)	93,3 (4,4s)
LDA (1kHz con características)	91,9 (6,5s)	155	95,5 (3,7s)

Tabla 2. Rendimiento de algoritmos de procesamiento de señales en personas amputadas. Tomado de: D. B. Alberto, G. Emanuele, C. Giorgio, D. Angelo, S. Rinaldo, G. Eugenio y Z. Loredana, «NLR, MLP, SVM, and LDA: a comparative analysis on EMG data from people with trans-radial amputation» Journal of NeuroEngineering and Rehabilitation, 2017

Este proyecto es clave ya que durante la ejecución del mismo se realiza un estudio similar para comparar los resultados obtenidos por distintos algoritmos de procesamiento de señales, pero a diferencia de este, los experimentos se realizan en personas sanas y se comparan con el algoritmo desarrollado con redes neuronales artificiales.

Así mismo se han llevado a cabo gran cantidad de proyectos relacionados, tales como controles de fuerza bio-inspirados, sistemas de control mediante el uso de redes neuronales artificiales, sistemas de control de agarre y deslizamiento para prótesis de miembro superior [11, 12, 13]. Sin embargo, la principal diferencia radica en la aplicación de las redes neuronales en un sistema que está dirigido a la rehabilitación de miembros.

3. JUSTIFICACIÓN

El hombre ha enfocado gran parte de sus energías en la investigación, principalmente en el campo de la medicina, los avances desarrollados hoy en día permiten a la humanidad tener calidad de vida a un grado razonable. Ahora bien, hay muchas personas que, debido a accidentes laborales, enfermedades u otros factores, pierden miembros de su cuerpo los cuales son reemplazados por prótesis que les permiten realizar diversas actividades.

A partir de la investigación se han encontrado formas de realizar el diseño de prótesis que permita al usuario el movimiento de algunas articulaciones mecánicas imitando los movimientos de la mano. En el diseño de éstas, es necesario el desarrollo de algoritmos que permitan procesar las señales recibidas de sensores mioeléctricos para controlar el mecanismo, imitando así de forma más natural el movimiento de la mano.

Con el propósito de ahondar en el campo de la inteligencia artificial aplicado a sistemas bio-inspirados, se propone el diseño de un algoritmo que sea capaz de reconocer con mayor exactitud los movimientos de la mano usando una red neuronal y un sensor Myo.

En muchos lugares se han llevado a cabo proyectos para diseñar algoritmos dirigidos al control de sistemas mecánicos para miembros superiores, pero son pocos los que usan sensores Myo, de hecho, la mayoría de los proyectos utilizan sensores intramusculares o electrodos de superficie. El sensor Myo tiene varias ventajas, una de ellas es que permite recibir los datos de forma inalámbrica, esta característica permite evaluar las señales EMG del brazo en diversas posiciones. Un estudio de dicha forma podría arrojar resultados útiles a la hora de observar los atributos de las señales que emiten los músculos en diferentes posiciones según el movimiento de los dedos de la mano. Otra ventaja que ofrece este sensor es el uso de sus 8 electrodos que se encuentran ubicados alrededor del brazo. Estos permiten recibir señales en tiempo real emitidas por cada músculo del antebrazo y, por lo tanto, obtener más información del movimiento realizado.

Por otra parte, aunque se tiene conocimiento de muchas formas de caracterizar las señales EMG e incluso proyectos en los cuales se han realizado experimentos para determinar cuáles algoritmos son más eficientes [10], hay que señalar que el uso de redes neuronales para el control de dichos sistemas puede ofrecer una mayor movilidad además de precisión en el reconocimiento de los movimientos de la mano ya que es posible entrenar una red con algunas características básicas de las señales y ésta a su vez podrá aprender a identificarlas con mayor facilidad relacionándolas con el movimiento indicado por el usuario.

Finalmente, uno de los principales objetivos del trabajo de investigación radica en ofrecer un avance en el área de inteligencia artificial, en cuanto a la aplicación de algoritmos de aprendizaje automático para el procesamiento de señales mioeléctricas, que forman parte de los grupos MEM y GED pertenecientes a la Facultad de Ingeniería Electrónica de la Universidad Santo Tomás.

4. OBJETIVOS

4.1 OBJETIVO GENERAL

Diseñar un algoritmo que mediante señales EMG sea capaz de identificar los movimientos de la mano y entrenar una red neuronal que permita mejorar el reconocimiento de los mismos.

4.2 OBJETIVOS ESPECÍFICOS

- Realizar un estudio acerca de la generación las señales mioeléctricas emitidas por el antebrazo a raíz de los movimientos de los dedos de la mano usando las bases de datos de la Universidad Santo Tomás (IEEE Explorer, Scopus, entre otros).

- Registrar señales EMG de ocho movimientos de los dedos de la mano usando el sensor Myo para adquirir la información y utilizarla en el procesamiento de las señales.

- Desarrollar un algoritmo en Matlab que utilice las señales EMG almacenadas para el entrenamiento de una red neuronal que permita reconocer los movimientos de la mano.

- Comparar el algoritmo desarrollado con un algoritmo estadístico, para determinar la precisión del reconocimiento con respecto a otras técnicas.

5. MARCO TEÓRICO

En este ítem se describen algunos conceptos necesarios para el uso y el desarrollo del algoritmo en cuestión.

5.1 MATLAB

Es un entorno informático numérico y un lenguaje de programación de secuencias de comandos. Los entornos informáticos numéricos como MATLAB permiten un trabajo matemático mucho más sofisticado que los programas de calculadora simples o los paquetes de software de hoja de cálculo. MATLAB es la abreviatura de "laboratorio de matrices", y su capacidad para trabajar con matrices fue una de sus fortalezas originales, aunque ahora es solo una de las muchas características. Originalmente pensado como un acceso directo para estudiantes que no conocían FORTRAN, ahora también interactúa con programas escritos en C, C ++, Java y Python, y permite a los usuarios crear sus propias interfaces de usuario [1].

5.2 REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales nacen como un intento del hombre de imitar la capacidad de procesamiento de información del cerebro humano. Alrededor de 1888 el médico español Santiago Ramon y Cajal propone la teoría de que las neuronas biológicas funcionan como entes independientes que se comunican entre sí a través de la sinapsis [2]. Con base en este concepto, empezaron a surgir algoritmos de procesamiento que establecen un paralelo entre la neurona biológica y la neurona artificial.

5.2.1 Neurona Biológica

Una neurona es una célula que, a diferencia de las demás células que componen nuestro cuerpo, tiene la capacidad de comunicarse con otras a través de señales eléctricas o químicas (Figura 5). Una neurona puede recibir señales de otras neuronas, combinarlas y emitir señales de salida. Este proceso lo puede realizar simultáneamente con miles de señales de entrada y producir miles de señales de salida. Las señales de entrada que acumula una neurona pueden ser modificadas por los pesos sinápticos y estas respuestas combinadas le permiten a la neurona tomar decisiones [2].

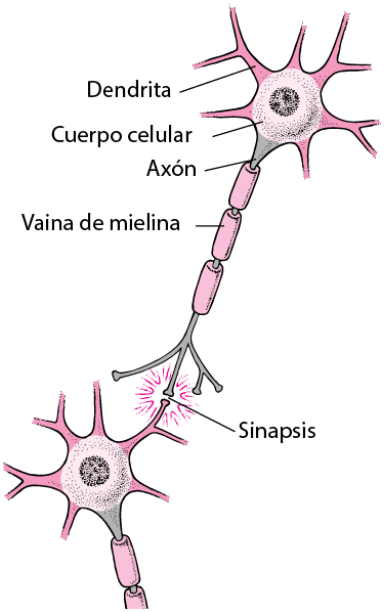

Figura 5. Estructura de una neurona biológica. Tomado de: https://www.msdmanuals.com/es/hogar/multimedia/figure/neu_nerve_cell_es

5.2.2 Neurona Artificial

Una neurona es la unidad básica de procesamiento de información. Esta unidad cuenta con variables de entrada que serán procesados dentro de la neurona y generarán un valor de salida. Se puede decir que la neurona es una función matemática que internamente realiza una suma ponderada de las entradas (x1, x2,…, xn) por los pesos (w1, w2,…, wn), es decir, por un valor que le asigna a cada entrada un grado de importancia en el cálculo. Los pesos resultan ser la parte fundamental de la neurona ya que ajustando los pesos la neurona puede aprender. El umbral suele marcar el límite entre una respuesta y otra ya que una sola neurona solo puede ofrecer respuestas binarias.

Se puede establecer un paralelo entre el funcionamiento de la neurona biológica y la neurona artificial (Figura 6):

1. **Entradas:** Tal como una neurona biológica puede obtener información de miles de neuronas a través del axón, una neurona artificial puede procesar información de miles de entradas.
2. **Sinapsis:** Una neurona biológica modifica los pesos sinápticos según las señales que recibe, de la misma manera, en una neurona artificial se ajustan los pesos según las entradas.
3. **Cuerpo:** El cuerpo de una neurona biológica combina las señales recibidas para emitir señales de salida, de forma paralela la neurona artificial realiza una suma ponderada de las entradas que recibe, multiplicadas por los pesos.

4. **Función de Activación:** El cuello del axón determina cuándo las señales que ha recibido la neurona cumplen las condiciones para emitir una señal de salida, en una neurona artificial se determina una función matemática que se ajuste de la mejor manera al problema que se debe solucionar.

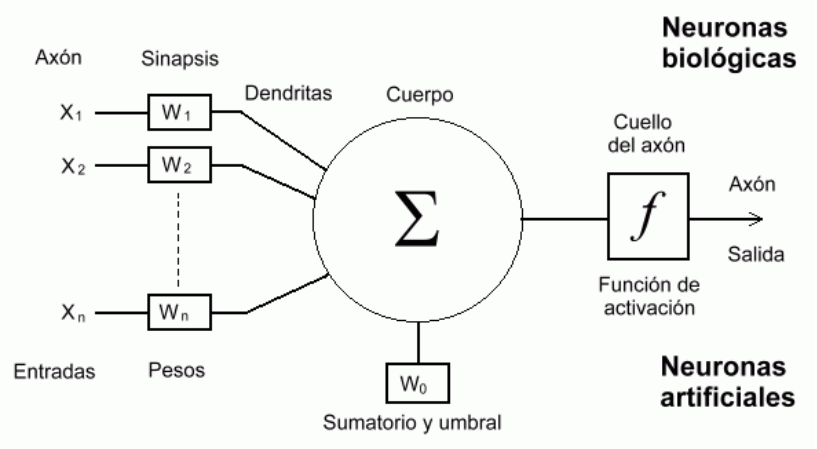

Figura 6. Estructura de una neurona artificial. Tomado de: http://www.cs.us.es/~fsancho/?e=72

5.2.3 Estructura de las Redes Neuronales

La estructura de una red neuronal es la forma en que se organizan las neuronas para procesar la información [2]. En este punto podemos clasificarlas en: Redes Monocapa y Redes Multicapa.

Las redes monocapa son aquellas que cuentan con una primera capa de neuronas que recibe las entradas, esta capa no realiza ningún tipo de procesamiento, por lo que no se considera formalmente una capa, luego son modificadas por los pesos sinápticos para llegar a una última capa que corresponde a las salidas de la red.

Las redes multicapa cuentan con una capa de neuronas de entrada y una de salida al igual que las redes monocapa, pero se diferencian en que este tipo de red neuronal posee capas ocultas de procesamiento.

5.3 ELECTROMIOGRAFÍA (EMG)

La electromiografía es una evaluación gráfica de los impulsos eléctricos emitidos por los músculos. Este tipo de evaluación es usada en medicina con el propósito de identificar posibles daños o lesiones musculares. Dado que la estructura muscular de cada persona es diferente a causa de la edad o sus propiedades musculares

genéticas, no existen protocolos para realizar un examen electromiográfico. Algunos principios generales involucran un estudio preliminar del paciente tomando muestras del músculo en reposo y con diferentes grados de contracción voluntaria, de esta manera se puede identificar la mejor posición para los electrodos [1].

La forma en que se generan las señales electromiográficas a nivel muscular permiten el monitoreo de estas. Cuando se genera un solo impulso neural este viaja a través de las fibras musculares unidas al mismo axón, así se genera un potencial en la unidad motora que se puede registrar fácilmente usando electrodos de superficie [1].

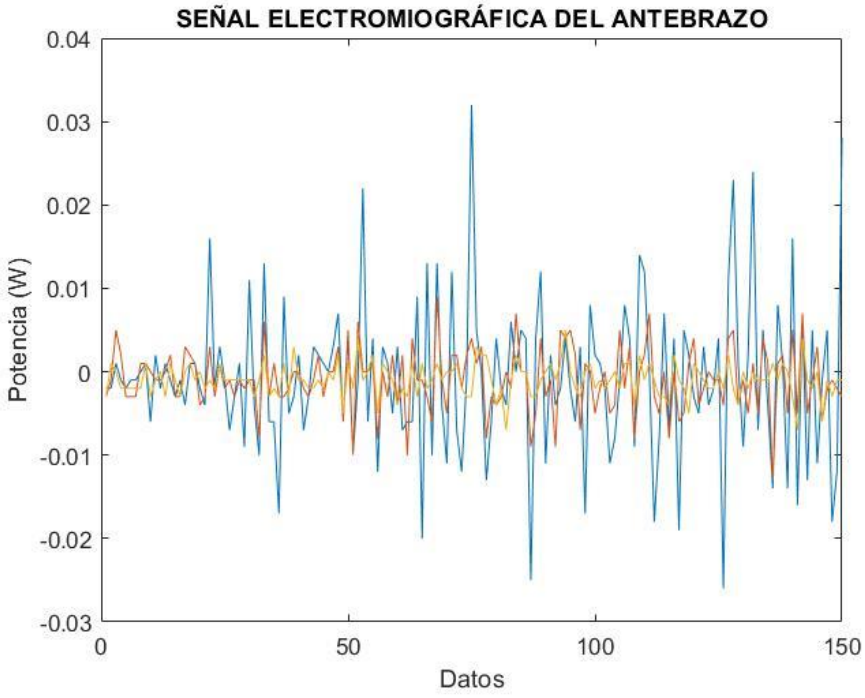

Figura 7. Ejemplo de una señal electromiográfica del antebrazo tomada con electrodos de superficie. Tomado de: Autor.

5.4 TRANSFORMADA DE FOURIER EN SEÑALES EMG

Las series de Fourier se han caracterizado por ser un sistema muy efectivo para analizar señales no periódicas, ya que permite su representación en el dominio de la frecuencia [14]. Dada la naturaleza no periódica de las señales EMG, este sistema

resulta una opción para analizar las diferencias entre las señales musculares emitidas por impulsos voluntarios.

Una de las propiedades de la transformada de Fourier es la densidad espectral de energía que proporciona información acerca de la cantidad de energía de una señal en el dominio de la frecuencia, esta propiedad permite clasificar señales a partir de la ubicación de los picos de energía, algo que no sería posible determinar en el dominio del tiempo [14].

Cuando un individuo genera un movimiento, las señales emitidas por los músculos presentan diferentes densidades espectrales que pueden ser usadas para clasificar el tipo de movimiento que se ha realizado. Un ejemplo de esto es la diferencia en el espectro de las señales de los dedos de la mano (figura 8).

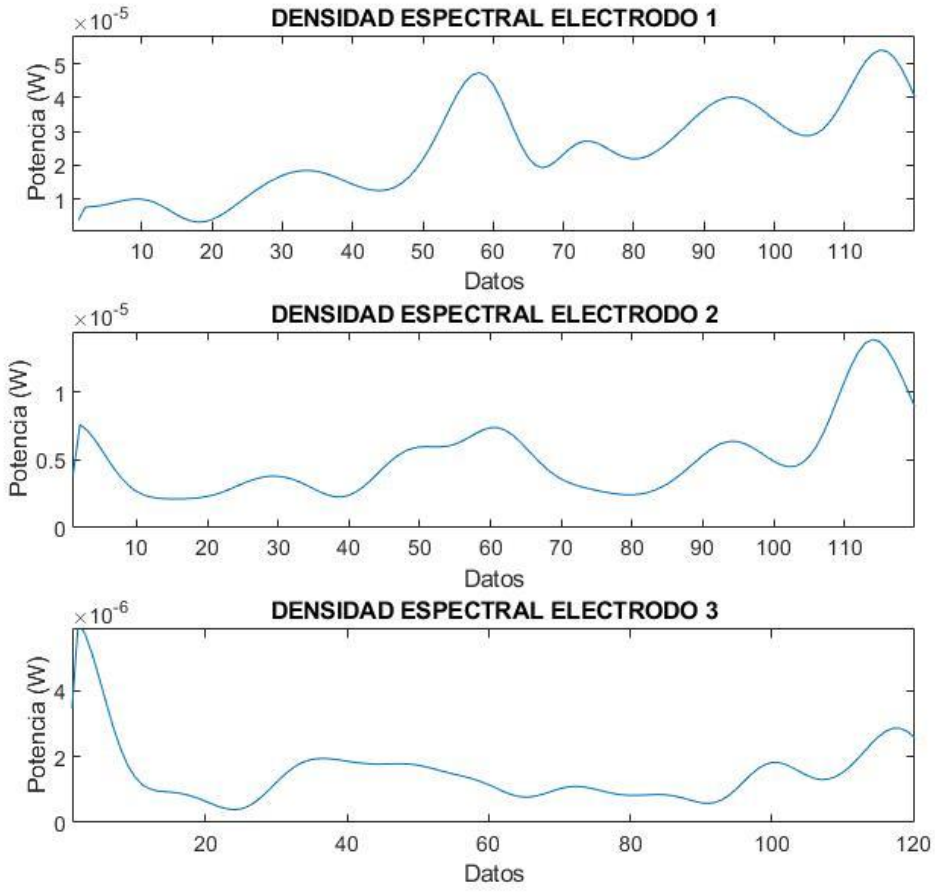

Figura 8. Densidad espectral de las señales EMG de 3 electrodos al mover el dedo índice. Tomado de: Autor.

5.5 ALGORITMO DE CRUCES POR CERO

El algoritmo de cruces por cero es un método usado ampliamente para detectar la frecuencia de las señales electrocardiográficas (ECG), ya que de forma simple se puede detectar el ritmo cardiaco [15]. Este algoritmo consiste en definir un umbral y comparar las muestras consecutivas para detectar cuando se encuentran en diferentes extremos del umbral.

$$Cruce\ por\ Cero = \frac{|x(n) - x(n-1)|}{2} \tag{1}$$

La cantidad de cruces por cero indican un valor cercano a la frecuencia natural de la señal que se está muestreando. En el caso de las señales electromiográficas (EMG), el valor del conteo de cruces por cero puede arrojar información de la frecuencia de los músculos involucrados, y por lo tanto ofrecer una posibilidad de clasificación de estos.

6. DISEÑO METODOLÓGICO

Una vez establecidos los conceptos necesarios para la comprensión del proyecto, es necesario llevar a cabo una serie de pasos que permitirán la ejecución del algoritmo y su correcto funcionamiento. Se da inicio con una breve descripción de los requisitos de software y hardware, luego se mencionan los pasos para la instalación de las librerías, el uso adecuado de los sensores, una descripción de las etapas en que está dividido el algoritmo para la identificación de los movimientos de la mano y, finalmente, se explica el funcionamiento de la interfaz de usuario.

6.1 MATERIALES

- IEEE Explorer
- Software MATLAB
- Sensor Myo

6.2 REQUERIMIENTOS

El uso y la implementación de los algoritmos de reconocimiento requieren un computador con MATLAB 2017 o superior ya que los componentes necesarios para su funcionamiento no están disponibles en versiones anteriores.

6.3 INSTALACIÓN

1. Descargar e instalar *Myo Connect*.*
2. Descargar el SDK y guardarlo en el disco local C.*
3. Agregar la dirección de la carpeta bin del SDK a las variables de entorno.
4. En MATLAB, instalar el Add-On MinGW-w64 C/C++ Compiler.
5. Instalar el Add-On Multi-layer Perceptron.

El programa debe ejecutarse desde su carpeta origen para que no ocurran errores.

6.4 MUESTREO

Se creó una base de datos con señales que permitieran entrenar la red neuronal artificial, para esto fue necesario determinar en qué posición debía estar el antebrazo para que la señal fuera lo más limpia posible. Con este fin se tomaron señales del brazo en reposo vertical y en reposo horizontal sobre una superficie (Figuras 9 y 10).

*La información necesaria para la instalación y el instalador se pueden obtener desde la página de soporte de Myo [18].

Figura 9. Brazo en posición vertical. **Tomado de: Autor.**

Figura 10. Brazo en posición horizontal sobre una superficie. **Tomado de: Autor.**

Tras determinar que la mejor posición para el muestreo es el brazo en posición vertical, se inició la toma de muestras de los dedos de forma individual. Para tomar las muestras, se ubica el sensor en el antebrazo, con el led en dirección a la región carpal anterior, como se contempla en la figura 11.

Figura 11. Ubicación adecuada del sensor Myo en el antebrazo. **Tomado de: Autor.**

Luego se sincroniza el sensor, y se espera unos minutos mientras aumenta la temperatura de los electrodos, de esta manera mejoran las muestras.
Para muestrear los datos, se desarrolló un programa en MATLAB el cual realiza la conexión con el sensor, ajusta el tiempo de muestreo y guarda los datos obtenidos.

La base de datos se amplió gradualmente en el proceso.

6.5 ALGORITMO

6.5.1 Tratamiento de datos

El proceso de muestreo de las señales se realiza en un intervalo de 5 segundos, como se puede observar en la figura 12a, el movimiento que realiza el usuario se visualiza en la parte de la señal que posee mayor potencial, el tiempo restante muestra las señales del brazo en posición de reposo. Para realizar el proceso de reconocimiento de las señales es necesario usar un sistema que identifique la fracción en la que el usuario realiza el movimiento. Para esto se define una ventana de 150 datos y se centra en el punto que posee los mayores picos. Para esto se usa la función *findpeaks* de MATLAB que reconoce los picos de la señal y se configura para que encuentre los picos más altos que se encuentren a máximo 8 datos de distancia entre ellos. Luego, usando la función *axis* se establece una ventana con el pico más alto en el centro, de esta manera se puede determinar cuando el usuario realiza el movimiento. Este tipo de filtrado se puede usar para el muestreo en tiempo real.

$$Muestra = [X_{max-74} \quad \cdots \quad X_{max} \quad \dots \quad X_{max+75}] \tag{2}$$

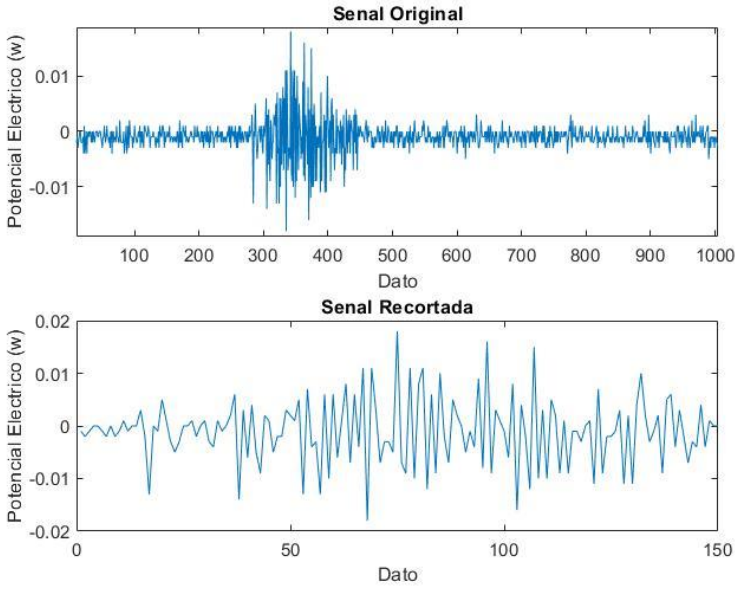

Figura 12. A. Señal dedo Corazón muestreada durante 5 segundos. B. Señal recortada a 150 datos. Tomado de: Autor.

Un punto importante es que no es relevante que el usuario se tome más de las 150 muestras para realizar el movimiento, puesto que al centrar la ventana dentro del lapso en el cual ocurre este, se obtiene el contenido frecuencial necesario para identificarlo. Sin embargo, no puede durar menos de 150 muestras, porque no arrojaría información suficiente para la identificación. Por esta razón se implementa también una alerta que le indique al usuario cuando debe repetir el ejercicio.

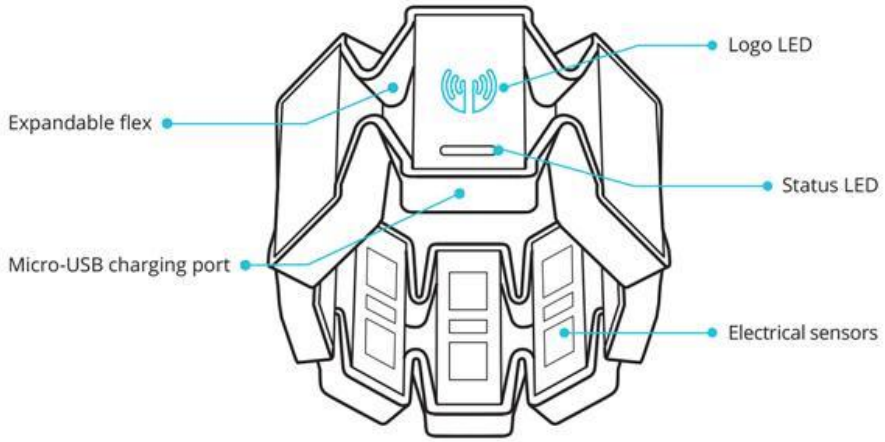

Figura 13. Estructura del sensor Myo. Tomado de: https://www.researchgate.net/figure/Myo-Armband-components-20_fig1_314079352.

El sensor Myo posee 8 electrodos, como se puede visualizar en la estructura de la figura 13, sin embargo, con el propósito de disminuir la cantidad de información que debe procesar el algoritmo, se identificaron los 3 electrodos que presentan mayor potencial cuando el usuario realiza el movimiento de algún dedo de la mano. En la figura 14 se encuentra la tabla comparativa de las señales de los 8 electrodos al realizar el movimiento del dedo pulgar. Estas señales muestran la transformada de Fourier de la señal que recibió cada electrodo para conocer el contenido frecuencial y determinar los electrodos con los que se obtienen mejores resultados. Tras observar las señales obtenidas se eligieron los electrodos 1, 2 y 3 para realizar el algoritmo.

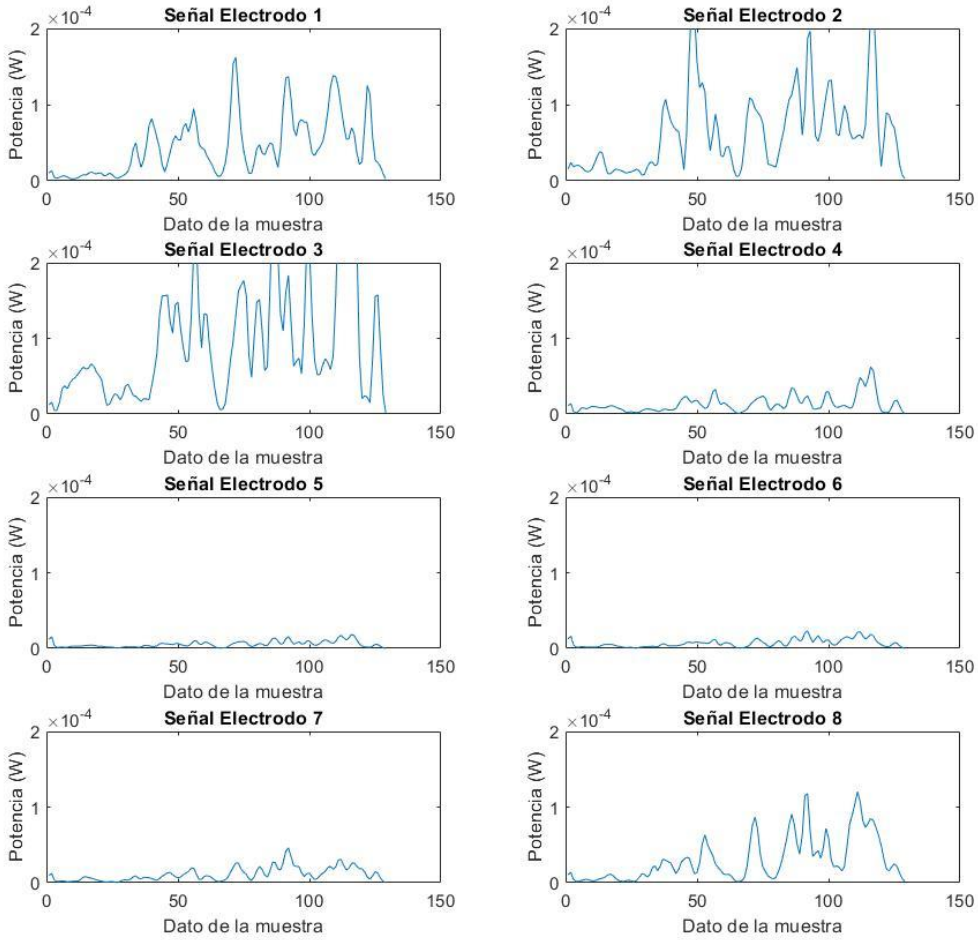

Figura 14. Señales del movimiento del dedo Pulgar para cada electrodo del sensor Myo. Tomado de: Autor.

6.5.2 Diseño de filtros

Para establecer las entradas de la red neuronal se filtran las muestras obtenidas, los filtros que se implementan son de tipo IIR por lo que incluyen una etapa de realimentación.

Los filtros consisten en un vector que recorre y se multiplica por la señal ingresada. El propósito del filtro es encontrar los componentes frecuenciales más comunes en cada muestra. Se implementa un filtro por cada tipo de movimiento que se quiere reconocer y por cada electrodo, es decir, si se evalúan 5 tipos de movimiento usando 3 electrodos se implementan 15 filtros.

Los filtros se basan en la estructura de una neurona artificial (ver figura 6), este sistema les permite adaptarse a las señales que recibe de entrada funcionando como un filtro multipasabandas.
Para calcular la salida de los filtros se realiza una multiplicación de la señal de entrada (X) por los pesos (W) que, inicialmente, serán valores aleatorios.

$$Y_{salida} = W_0 + \sum_{n=1}^{n} (W_n * X_n) \quad (3)$$

Luego, se calcula el error con base en la señal de salida deseada

$$Error = Y_{deseada} - Y_{salida} \quad (4)$$

Finalmente, se realiza un ajuste de los pesos de los filtros según el error calculado. Para esto se suma el peso obtenido con la multiplicación del error, la señal de entrada y $\propto$, un factor que compensa los cambios entre cada etapa del entrenamiento y que, por parámetros de diseño, se estableció en 0.001.

$$W = W + (\propto * Error * X) \quad (5)$$

Para el ajuste inicial de los filtros, se cargan las señales del dataset por cada electrodo y por cada muestra de forma consecutiva, luego se suman entre ellas y se aplica el filtro correspondiente a cada dedo. El resultado de este primer paso es un vector de pesos que se usa para comparar las siguientes muestras.

Para las nuevas muestras se cargan los pesos ajustados para cada filtro y se ingresa la señal a través de cada uno de ellos. El resultado de cada filtro es un número que varía entre 0 y 1, que corresponde al potencial del dedo al pasar por el filtro. Entre mayor similitud tenga la señal ingresada con respecto a los pesos del filtro, mayor es el potencial. El proceso realizado por el algoritmo de filtrado se detalla en la figura12, donde la función “Aplicar Filtros” corresponde al proceso descrito en las ecuaciones 3, 4 y 5.

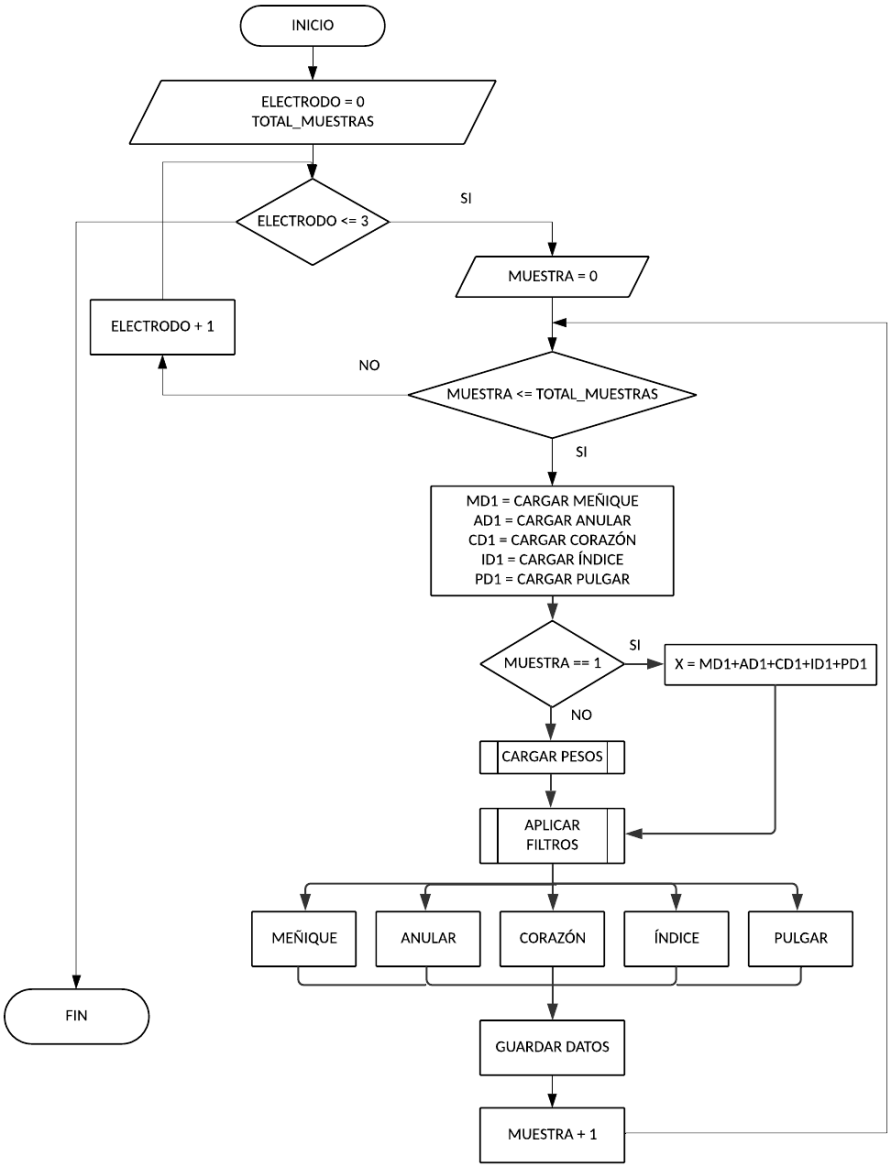

Figura 12. Diagrama de flujo del algoritmo usado para filtrar. Tomado de: Autor.

El algoritmo de entrenamiento establece dos valores iniciales, ω un vector aleatorio que representa los pesos y $\propto$. Luego, dedo a dedo, se simula la salida deseada multiplicando el valor de los pesos por el valor de los datos de la señal de entrada. Se calcula el error y se corrigen los pesos durante la cantidad de épocas que se indiquen, 2000 en este caso particular. Al finalizar el entrenamiento con los datos de un dedo, se guardan los valores de los pesos para realizar el filtrado (Figura 13).

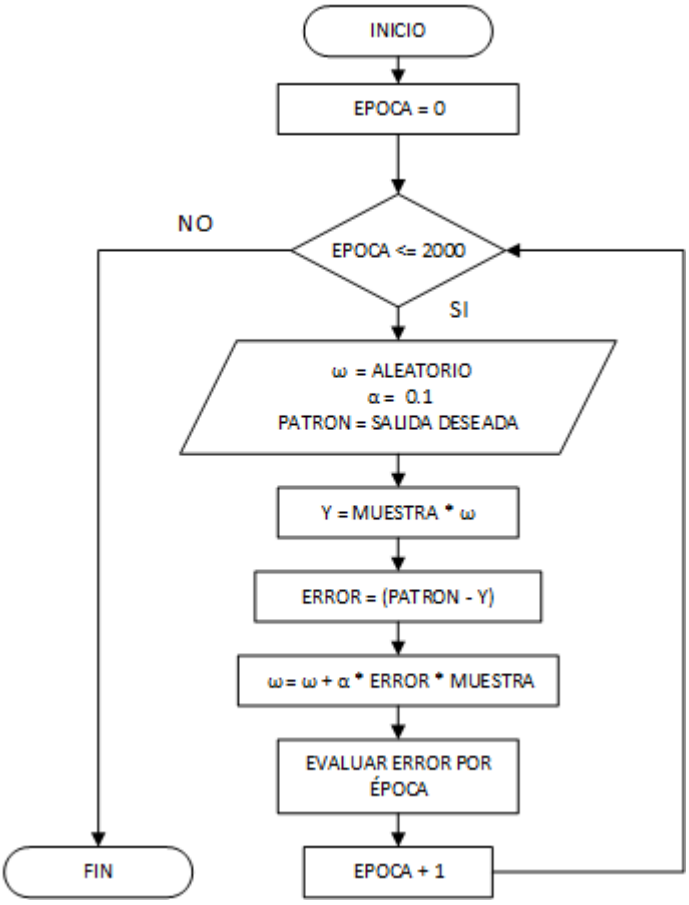

Figura 13. Diagrama de flujo del algoritmo de entrenamiento.
Tomado de: Autor.

Finalmente, se toman los pesos hallados en la etapa de entrenamiento y se multiplican por una señal de la base de datos, dado que se implementa un filtro por cada dedo, al final se obtienen 7 potenciales. El proceso se repite para cada muestra por dedo, en total se obtiene una salida con 21 potenciales que serán las entradas del perceptrón multicapa. Este proceso se detalla en la figura 14.

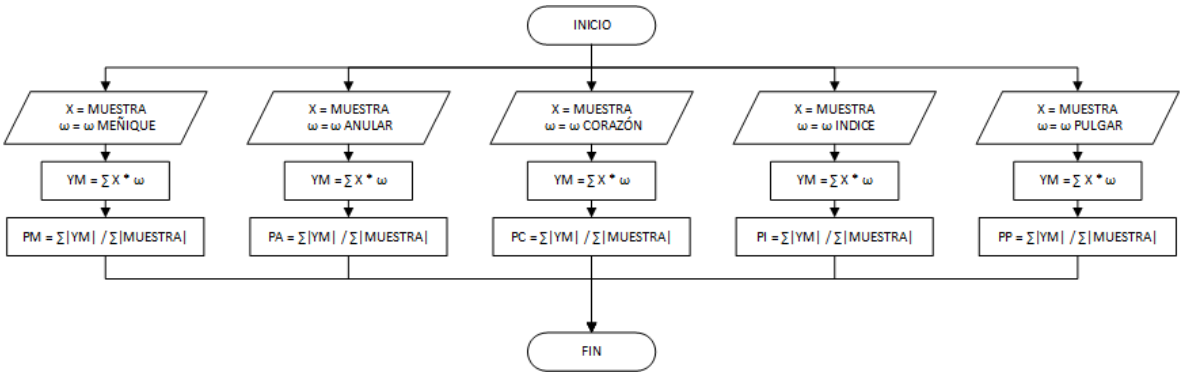

Figura 14. Diagrama de flujo de los filtros. Tomado de: Autor.

6.5.3 Red Neuronal Artificial

La red neuronal está compuesta por 3 fases: 1) Entradas, 2) Capas ocultas y 3) Salidas (Figura 15). En la primera fase se encuentran las salidas de los filtros, que son 7 por cada electrodo, lo que resulta en 21 entradas. La segunda fase son las capas ocultas que se configuran desde la librería de MATLAB. Y la tercera fase son las salidas, que son binarias, la única que se encuentre en 1 es la que indica el dedo que movió el usuario.

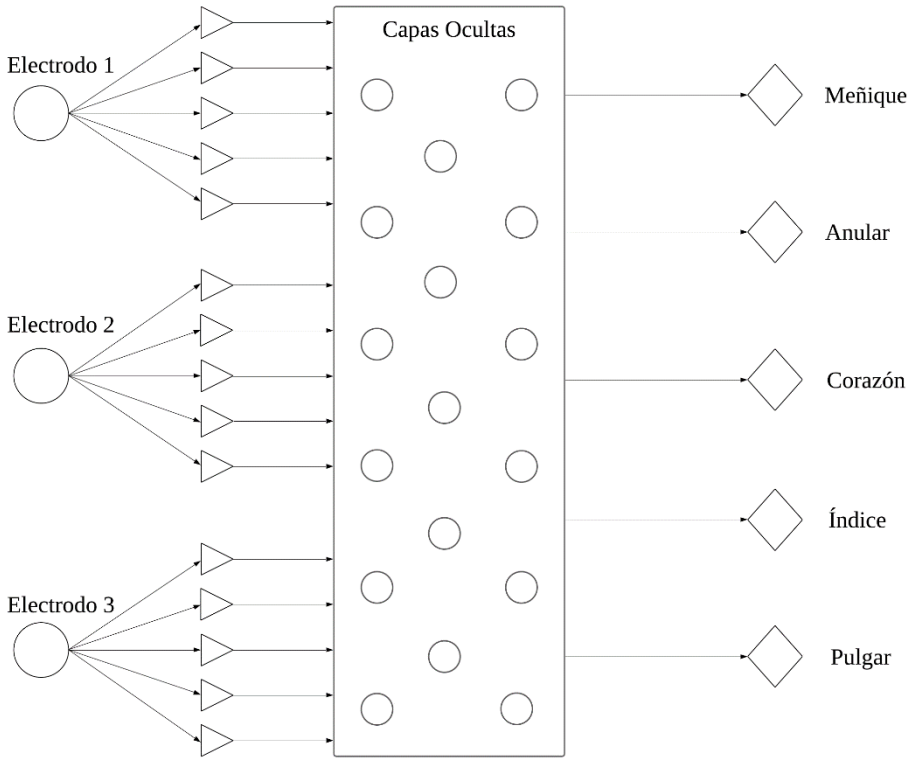

Figura 15. Diseño de la red neuronal artificial. Tomado de: Autor.

Para el entrenamiento se usa la librería Multi-Layer Perceptron que permite establecer los parámetros de entrenamiento tales como la cantidad de capas ocultas, la cantidad de neuronas por capa, la función de activación, el mínimo error deseado, entre otros.

La librería usada para el entrenamiento se configuró inicialmente con la función de activación de gradiente descendente. Esta función calcula las derivadas parciales de la función para encontrar los mínimos de esta:

$$Gradiente\ Descendente = \nabla f = \frac{\partial f}{\partial w_i}, para\ i = 1, \dots, n \tag{5}$$

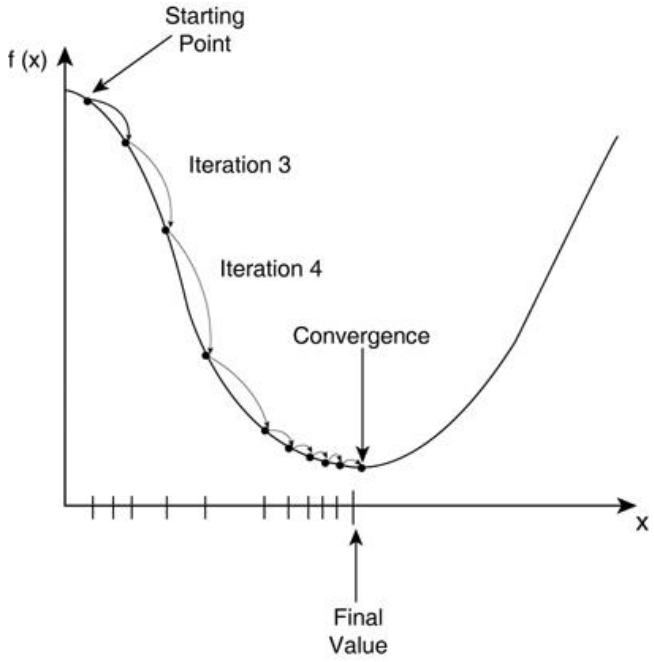

Figura 16. Disminución del error usando como función de activación el gradiente descendente. Tomado de: http://www.cs.us.es/~fsancho/images/2017-02/gds.gif.

Un buen entrenamiento usando el algoritmo de gradiente descendente debería arrojar una disminución gradual del error hasta conseguir el error más bajo posible, similar al de la figura 16. Sin embargo, esta función de activación no proporcionó buenos resultados para el algoritmo de reconocimiento, como se observa en la figura 17, el error disminuye y eventualmente se eleva nuevamente a causa de que la función no consigue establecer los mínimos.

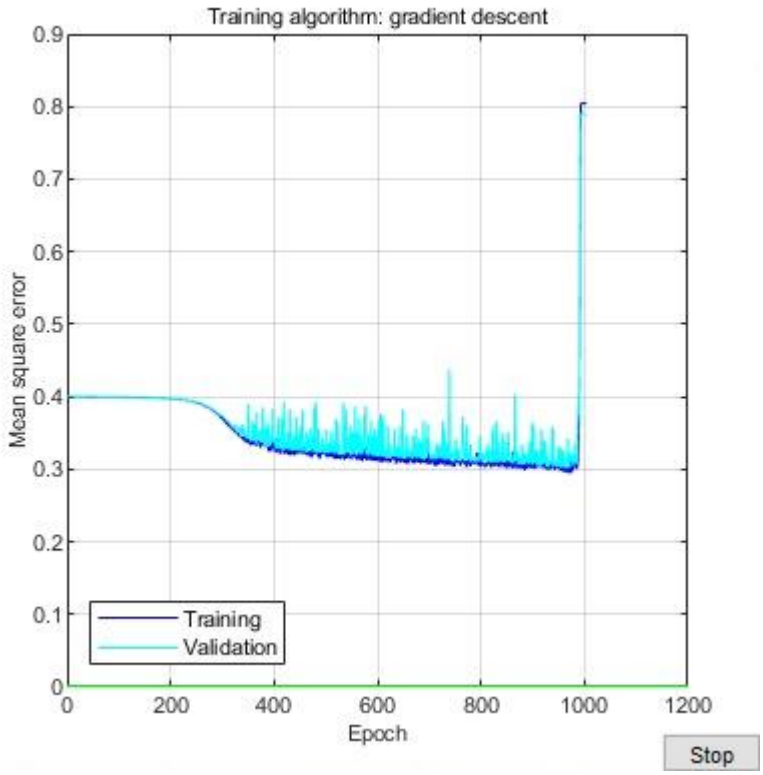

Figura 17. Resultado del entrenamiento usando Gradiente Descendente. Tomado de: Autor.

Como alternativa al algoritmo de gradiente descendente se tiene el algoritmo de Levenberg-Marquardt. Este algoritmo combina la teoría de gradiente descendente con el algoritmo de Gauss-Newton, de manera que busca los mínimos cuadrados de una señal sin usar segundas derivadas. Esta diferencia hace que el algoritmo sea más versátil ya que le permite ajustarse a las curvas y encontrar mínimos que no sean necesariamente globales [16]. Usando este algoritmo se obtiene como resultado la señal de la figura 18, donde se observa una disminución gradual del error y mayor estabilidad en la validación durante el entrenamiento. La validación de la que se habla es la respuesta del algoritmo a señales con las cuales no haya sido previamente entrenado.

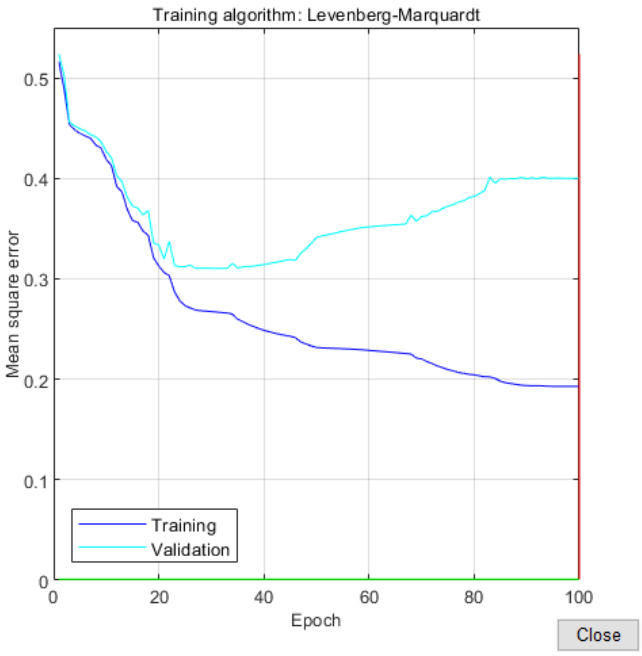

Figura 18. Resultado del entrenamiento usando Levenberg-Marquardt. Tomado de: Autor.

6.6 ALGORITMO ESTADÍSTICO

Para realizar la comparación de los resultados obtenidos por el algoritmo de aprendizaje automático, se implementó un algoritmo estadístico que cuenta la cantidad de cruces por cero de las señales de los tres electrodos.

Al clasificar las señales para identificar el movimiento que ha realizado el usuario, el algoritmo compara la señal obtenida de los tres electrodos con un promedio de la cantidad de cruces por cero de las señales del dataset. Este proceso se realiza encontrando la distancia euclidiana entre los cruces por cero, donde la menor distancia corresponde al movimiento más similar al que ha realizado el usuario. La distancia se calcula así:

$$Distancia\ Euclidiana = \sqrt{(P1 - M1)^2 + (P2 - M2)^2 + (P3 - M3)^2} \quad (6)$$

Donde P(n) es el promedio de cruces por cero de las señales del dataset, M(n) es la cantidad de cruces por cero de la muestra obtenida y n es el número del electrodo.

6.7 INTERFAZ GRÁFICA

La interfaz gráfica se implementó con el diseño que se observa en la figura 19, donde se presentan 3 botones:

- ENTRENAR: Este botón ejecuta el algoritmo de Multi-Layer Perceptron para la identificación de las señales muestreadas. Este botón habilita la opción de grabar, ya que es indispensable que se corra el algoritmo de entrenamiento antes de intentar identificar una señal. Cuando se ejecuta el algoritmo se puede visualizar el proceso del entrenamiento (Figura 20).
- GRABAR: Este botón ejecuta el programa de conexión con el sensor Myo y adquiere las señales. El botón cambia a color rojo cuando se está preparando para el muestreo y cambia a color verde cuando está grabando el movimiento realizado. El tiempo de muestreo es de 5 segundos. Una vez recibida la señal, pasa por los filtros y se envía a los algoritmos de la red neuronal y el algoritmo estadístico para realizar el respectivo proceso de identificación. Una vez finalizado se observa un mensaje que indica el resultado obtenido por cada uno de los algoritmos.
- GUARDAR: Este botón toma el valor de la última señal recibida por el sensor, el valor indicado en el contador “Muestra” y el item marcado en el recuadro “Dedos” para agregar información al dataset.

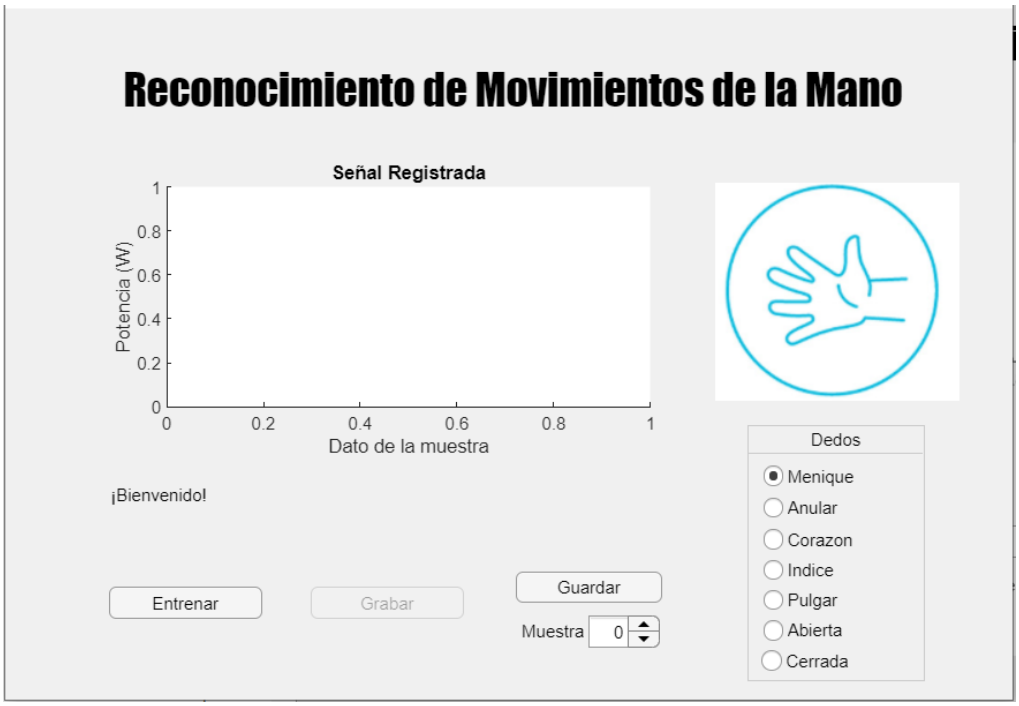

Figura 19. Interfaz gráfica. Tomado de: Autor.

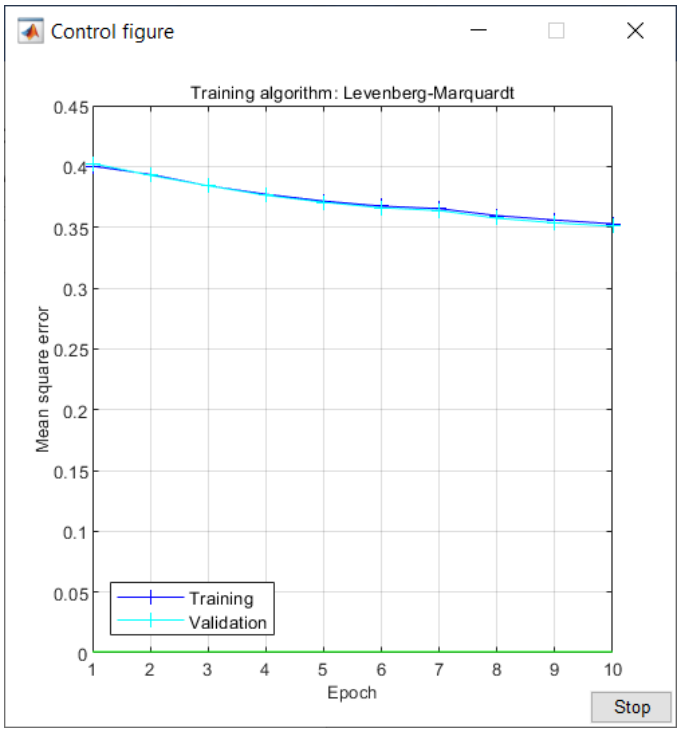

Figura 20. Ventana de entrenamiento. Tomado de: Autor.

7. RESULTADOS

7.1 DISEÑO DE EXPERIMENTOS

Se puede considerar que la mejor posición para muestrear una señal muscular es aquella en la que los músculos generen la menor cantidad de impulsos involuntarios, por pequeños que sean. A fin de determinar la mejor posición para tomar los datos del brazo en reposo, se realizó un experimento en el cual se tomaron datos del brazo en reposo horizontal sobre una superficie y vertical. Primero se tomaron los datos mientras el usuario movía un dedo, la señal se puede observar en las figuras 21 y 22, sin embargo, el resultado no fue concluyente, por lo cual se realizó otra toma con el brazo en reposo total durante el muestreo como se observa en las figuras 23 y 24. En este último experimento se puede observar que la señal tomada con el brazo en reposo vertical tiene menos impulsos involuntarios que la señal tomada con el brazo en reposo horizontal.

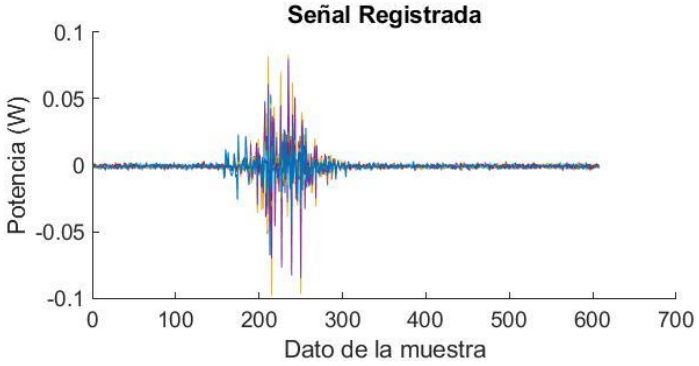

Figura 21. Señal del brazo en reposo vertical con movimiento de un dedo. Tomado de: Autor.

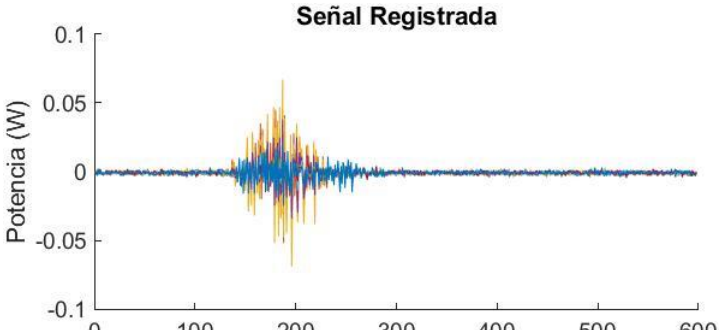

Figura 22. Señal del brazo en reposo horizontal sobre una superficie con movimiento de un dedo. Tomado de: Autor.

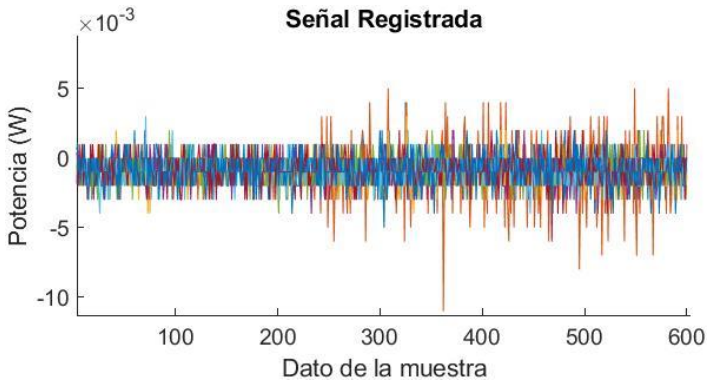

Figura 23. Señal del brazo en reposo horizontal sobre una superficie. Tomado de: Autor.

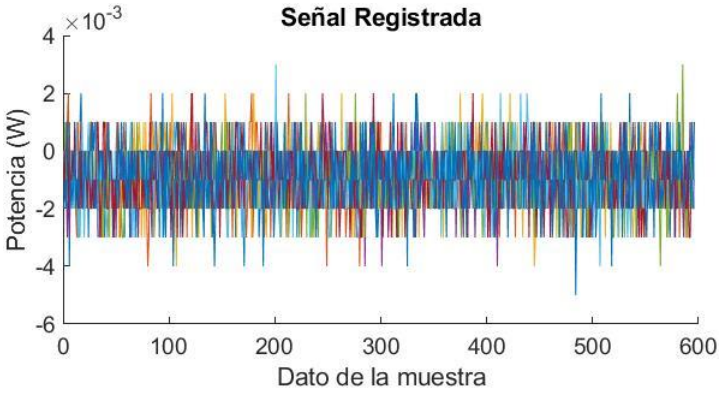

Figura 24. Señal del brazo en reposo vertical. Tomado de: Autor.

Las señales de la figura 24 muestran un potencial que varía entre 1mW y -1mW, con ligeras variaciones cerca de los 2mW, mientras en la figura 23 se observan picos mucho más pronunciados, algunos hasta superan los -5mW. Esto lleva a la conclusión de que el brazo en reposo vertical es la mejor posición para realizar el muestreo.

7.2 ALGORITMO DE MUESTREO Y ACONDICIONAMIENTO DE SEÑALES

El muestreo de la señal se realiza a 200Hz ya que es la frecuencia en la cual trabaja el sensor Myo. Para el funcionamiento del algoritmo es indispensable haber guardado el sdk en el disco local C y tenerla como variable de entorno, de lo contrario no se podrá conectar el sensor.

Primero se ejecutan las funciones de instalación de la librería Myo_Mex que permiten el acceso a la información del sensor. Luego se definen los datos de conexión del dispositivo. La propiedad countMyos indica la cantidad de sensores que se van a muestrear simultáneamente y, con los métodos startStreaming y

stopStreaming se inicia y detiene el muestreo. Se determina un intervalo de 3 segundos durante los cuales el sensor estará activo y el usuario deberá realizar el movimiento. Al terminar el tiempo, se obtienen las señales de los 3 primeros electrodos usando la propiedad emg_log. Luego se identifican los picos de la señal usando la función findpeaks indicando que los picos deben estar a una distancia mínima de 8 muestras. Una vez definidos los picos se identifica el valor del pico máximo y luego su ubicación en el vector, de esta manera se puede definir una ventana de 150 datos con la mayor cantidad de información de la señal. Finalmente, se destruye el sensor declarado para liberar el puerto.

```
%% INSTALACION DE MYO_MEX
install_myo_mex;
sdk_path = 'C:\myo-sdk-win-0.9.0';
build_myo_mex(sdk_path);

%% RECIBIR DATOS DE CONEXION DEL DISPOSITIVO
countMyos = 1;
m1 = MyoMex(countMyos);
m = m1.myoData;
pause(1);
m.clearLogs()
m.startStreaming();
pause(3);
m.stopStreaming();

signalE1 = m.emg_log(:,1);
signalE2 = m.emg_log(:,2);
signalE3 = m.emg_log(:,3);
[pks,locs] = findpeaks(signalE1,'MinPeakDistance' , 8);

%% UBICACION DEL PICO MAXIMO
M=max(pks);
for N=1:length(pks)
    if M == pks(N)
        M = N;
    end
end

%% DEFINIR LA VENTANA SEGÚN EL PICO MÁXIMO
% Se toman 150 datos con el punto maximo en medio.
snewE1=signalE1((locs(M))-74:(locs(M))+75);
snewE2=signalE2((locs(M))-74:(locs(M))+75);
snewE3=signalE3((locs(M))-74:(locs(M))+75);

%% FINALIZAR LA CONEXIÓN
m1.delete();
```

El resultado de la ejecución de este algoritmo son 3 señales, que corresponden a los 3 electrodos seleccionados para este desarrollo, y sus 3 muestras recortadas de 150 datos cada una. Se puede obtener la gráfica de estas señales con las siguientes líneas:

```
% PINTAR LA SEÑAL ORIGINAL Y LA RECORTADA
hold on
subplot(2,1,1),plot(signalE1), title('Senal
Original'),xlabel('Dato'),ylabel('Potencial Electrico (w)')
subplot(2,1,2),plot(snewE1), title('Senal
Recortada'),xlabel('Dato'),ylabel('Potencial Electrico (w)')
hold off
```

La gráfica resultante será similar a las de la figura 25.

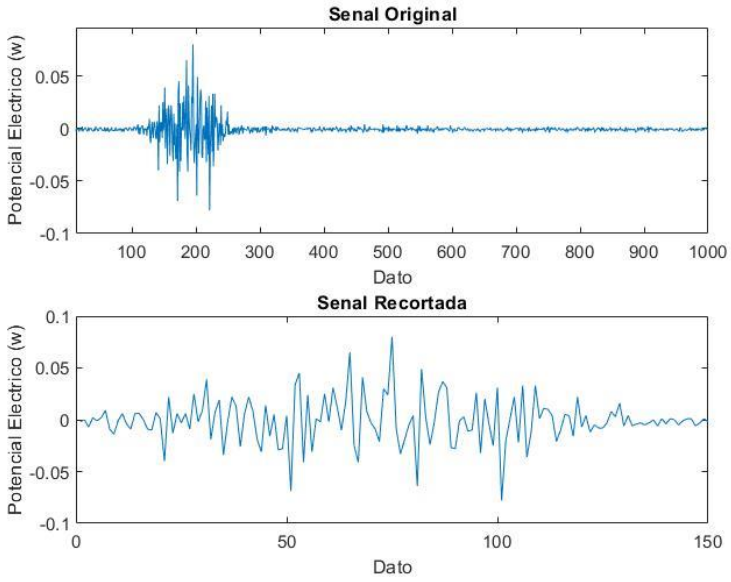

Figura 25. Señal dedo Pulgar recortada a 150 datos. Tomado de: Autor.

7.3 FILTROS

Cuando se carga la primera muestra de cada dedo se ejecuta el algoritmo de entrenamiento; este algoritmo corrige gradualmente los pesos (ω) para ajustar los filtros según el dedo que se desee identificar. En la figura 26 se observa la disminución gradual del error al ajustar el filtro de una muestra del dedo meñique. Cabe notar que el error no llega a cero, esto se debe a que el algoritmo tiene limitaciones, de manera que no puede aprender a la perfección el comportamiento de la señal.

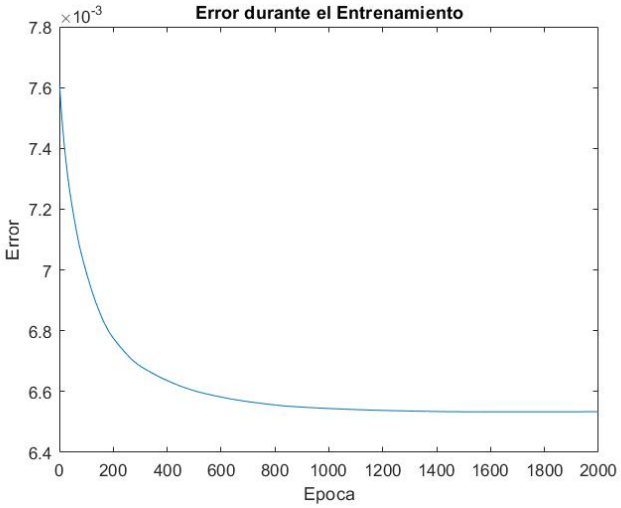

Figura 26. Disminución del error durante el entrenamiento de una muestra del dedo meñique. Tomado de: Autor.

Cuando se ha ajustado el filtro, la señal de salida tiende a parecerse a la señal de entrada, este comportamiento demuestra que el sistema está aprendiendo el comportamiento de la señal. En la figura 27 se puede notar que la señal real toma como guía los picos más pronunciados de la señal a la que debe parecerse.

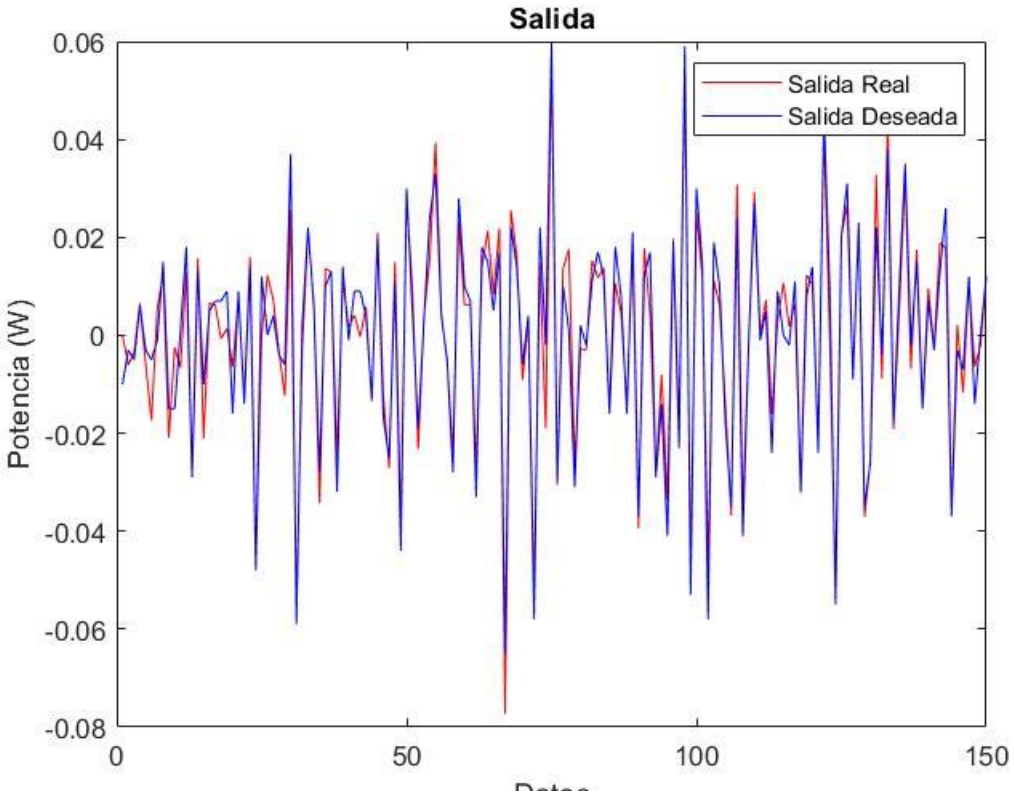

Figura 27. Señal de salida tras el entrenamiento vs. señal de entrada. Tomado de: Autor.

Al finalizar el entrenamiento y el filtrado de las señales, se tiene a la salida 7 potenciales por cada dedo, de manera que si se entrenaran los filtros con 5 muestras se obtendría una tabla con una estructura similar a la que se observa en la Tabla 3 para cada dedo. La tabla 3 corresponde a la salida del entrenamiento con 5 muestras del dedo anular del electrodo 1.

MUESTRA	MENIQUE	ANULAR	CORAZON	INDICE	PULGAR	ABIERTA	CERRADA
1	0.0158	0.0157	0.0156	0.0152	0.0147	0.0155	0.0154
2	0.0181	0.0186	0.0183	0.0174	0.0177	0.0182	0.0179
3	0.0173	0.0174	0.0172	0.0191	0.0185	0.0188	0.0186
4	0.0175	0.0183	0.0187	0.0177	0.0185	0.0184	0.0180
5	0.0178	0.0188	0.0187	0.0172	0.0193	0.0185	0.0187

Tabla 3. Salida con 5 muestras del dedo anular (Electrodo 1).

Para obtener los valores que ingresarán a la red neuronal se concatenan los vectores de salida por electrodo como se ilustra en la Tabla 4.

ELECTRODO 1							ELECTRODO 2							ELECTRODO 3						
M	A	C	I	P	AB	CE	M	A	C	I	P	AB	CE	M	A	C	I	P	AB	CE
0.24	0.23	0.23	0.24	0.24	0.23	0.23	0.22	0.21	0.21	0.19	0.21	0.21	0.21	0.15	0.09	0.09	0.11	0.09	0.09	0.11
0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02
0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02
0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02
0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.02	0.03	0.02	0.02	0.02	0.02

Tabla 4. Datos de entrada para la red neuronal con 5 muestras del dedo anular. Tomado de: Autor.

7.4 RED NEURONAL ARTIFICIAL

La librería Multi-Layer Perceptron se encarga de implementar el algoritmo para el entrenamiento de la red con los parámetros iniciales, que en este caso se configuraron de la siguiente manera:

```
MLP = ...
MultiLayerPerceptron('LengthsOfLayers', [21 7 7],...
'HiddenActFcn',      'tanh',...
'OutputActFcn',      'softmax');
```

Donde el parámetro LengthsOfLayers corresponde a las capas internas de la red neuronal, HiddenActFnc es la función de activación de entrada y OutputActFnc es la función de activación de salida. ‘

Se configura el algoritmo con una capa inicial de 21 neuronas que corresponde a las 21 entradas del algoritmo y dos capas de salida de 7 neuronas que corresponden a las 7 posibles salidas (Meñique, Anular, Corazón, Índice, Pulgar, Abierta, Cerrada). Para el caso de la función Softmax es importante que las últimas dos capas tengan la misma cantidad de neuronas.

Adicional, se configuran el entrenamiento para 100 épocas, un error mínimo de $1x10^{-3}$, y se configura el entrenamiento con el algoritmo de Levenberg-Marquardt.

```
Options = ...
    struct('TrainingAlgorithm', 'LM',...
           'NumberOfEpochs',    100,...
           'MinimumMSE',        1e-3,...
           'InitialMarquardtParameter', 0.01,...
           'MaximumMarquardtParameter', 1e10,...
           'MarquardtFactor',           10,...
           'BayesianRegularization',    false);
```

El resultado del entrenamiento muestra cómo el error baja cada época al igual que la validación hasta el 0.2, después de eso se observa cómo la validación incrementa mientras el error disminuye (Figura 28).

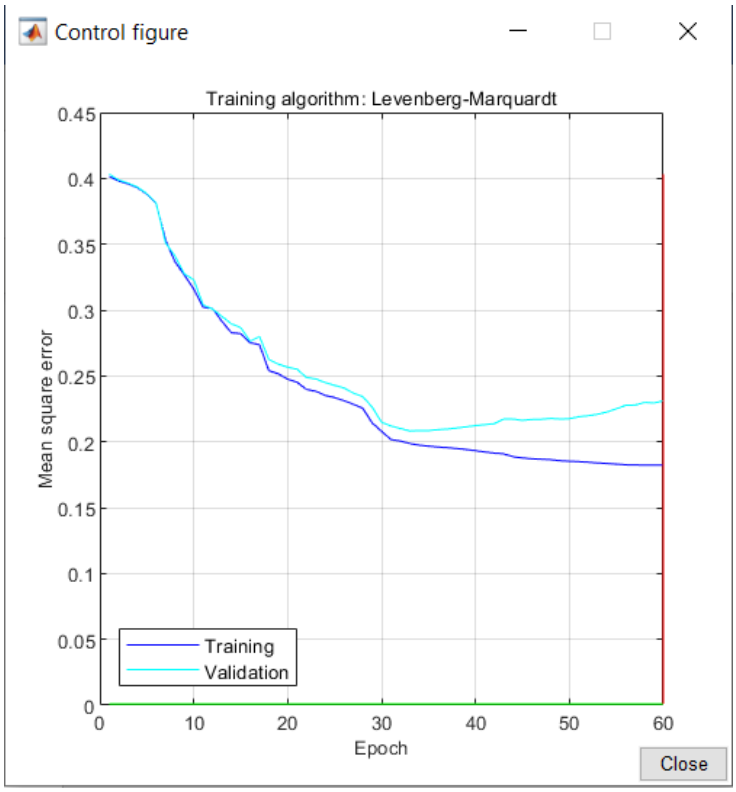

Figura 28. Error vs. Época durante el entrenamiento de la Red Neuronal. Tomado de: Autor.

Este comportamiento se debe a que el algoritmo al llegar aproximadamente a la época 30 se empieza a sobre entrenar, es decir, no aprende patrones de las señales sino las señales del dataset. Dado que el propósito es que el algoritmo aprenda los patrones de las señales de los dedos para que pueda identificar nuevas señales, es necesario que el entrenamiento no llegue a la etapa de sobre entrenamiento, por lo que se debe ajustar a la cantidad de épocas y demás parámetros que sean necesarios.

Teniendo presente el resultado anterior, se configura el algoritmo a 40 épocas obteniendo el resultado de la figura 29. Para el dataset con 300 muestras de cada movimiento, un total de 2100 muestras, el error cuadrático medio es del 20%.

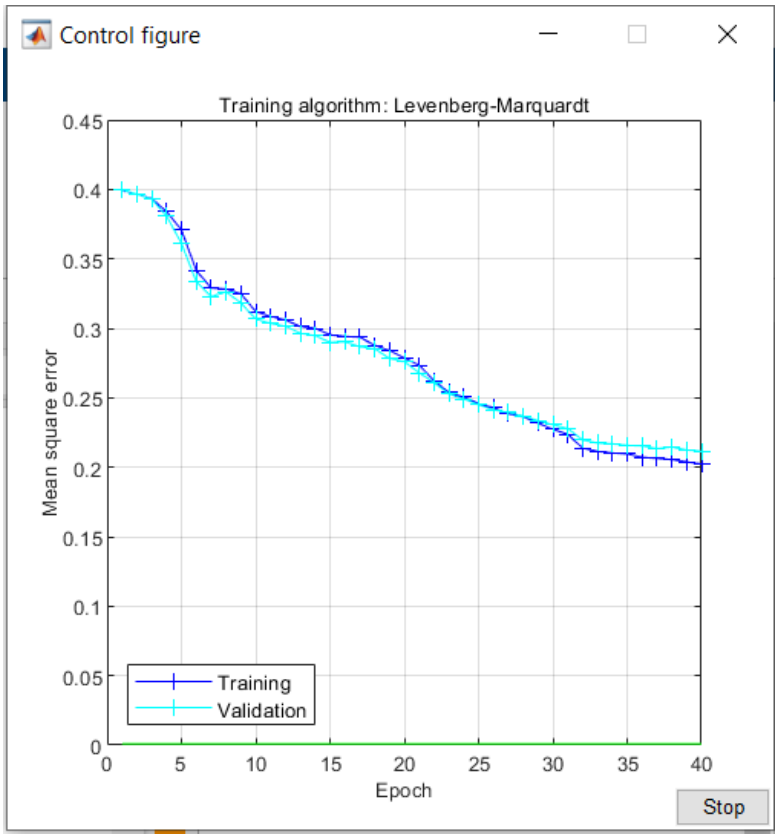

Figura 29. Resultado del entrenamiento con 40 épocas. Tomado de: Autor.

7.5 ALGORITMO ESTADÍSTICO

Para el algoritmo de conteo de cruces por cero se desarrolló una función que evalúa las señales punto a punto incrementando un contador cada vez que la señal pasa de positivo a negativo. El conteo se realiza de la siguiente manera:

```
ceros=[];
for i=1:(length(MD1)-1)
    if (MD1(i)*MD1(i+1))==0
        if MD1(i)==0
            ceros=[ceros;i];
        else
            ceros=[ceros;i+1];
        end
    elseif (MD1(i)*MD1(i+1))<0
        if abs(MD1(i))<abs(MD1(i+1))
            ceros=[ceros;i];
        else
            ceros=[ceros;i+1];
        end
    end
end
```

Donde *MD1* es la señal de uno de los electrodos y *ceros* es el arreglo que guarda las posiciones de los cruces por cero encontrados en la señal.

El promedio de cruces por cero obtenido de las señales del dataset se pueden observar en la Tabla 5.

	ELECTRODO 1	*ELECTRODO 2*	*ELECTRODO 3*
MEÑIQUE	95.4700	92.9367	90.5933
ANULAR	98.0867	90.3300	88.4567
CORAZÓN	97.4300	95.2033	97.6067
ÍNDICE	97.1467	92.4267	90.4067
PULGAR	98.3767	95.8167	95.2600
ABIERTA	95.2433	96.9267	97.7900
CERRADA	94.9000	92.7900	93.6067

Tabla 5. Promedio de cruces por cero con 300 muestras de cada movimiento. Tomado de: Autor.

7.6 COMPARACIÓN ENTRE ALGORITMOS

Para la comparación de los algoritmos se tomaron 350 muestras y se registraron los datos que se pueden observar en las Figuras 30 y 31.

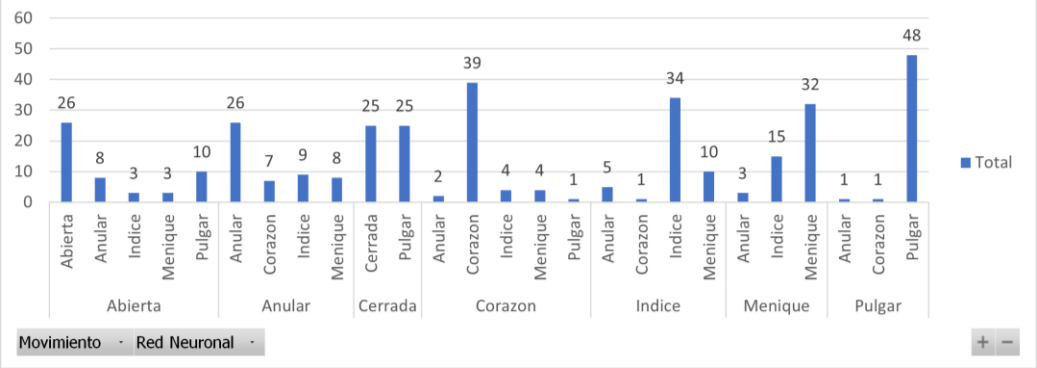

Figura 30. Resultados Red Neuronal. Tomado de: Autor.

La Red Neuronal acertó 232/350 lo que le da un porcentaje de acierto del 66.28% mientras el algoritmo estadístico acertó 110/350 mostrando un porcentaje de acierto del 31.42%.

Los resultados de esta prueba muestran que la Red Neuronal tiende a acertar aproximadamente 6 de cada 10 intentos, donde el movimiento más fácil de identificar es el dedo Pulgar.

Respecto al algoritmo estadístico se puede resaltar que el dedo corazón y el dedo pulgar tienen una frecuencia natural similar, razón por la cual el algoritmo de cruce por cero no distingue fácilmente la diferencia entre ellos.

En la tabla 6 se presenta la matriz de confusión, esta tabla permite determinar el rendimiento obtenido con los dos algoritmos desarrollados. De estos resultados podemos concluir que el promedio de precisión del algoritmo usando la red neuronal es del 90.36%, y predice correctamente los resultados un 66% de las veces.

Por el contrario, el algoritmo estadístico indica un promedio de precisión del 80.4% pero solo predice correctamente los resultados el 31.42% de las veces.

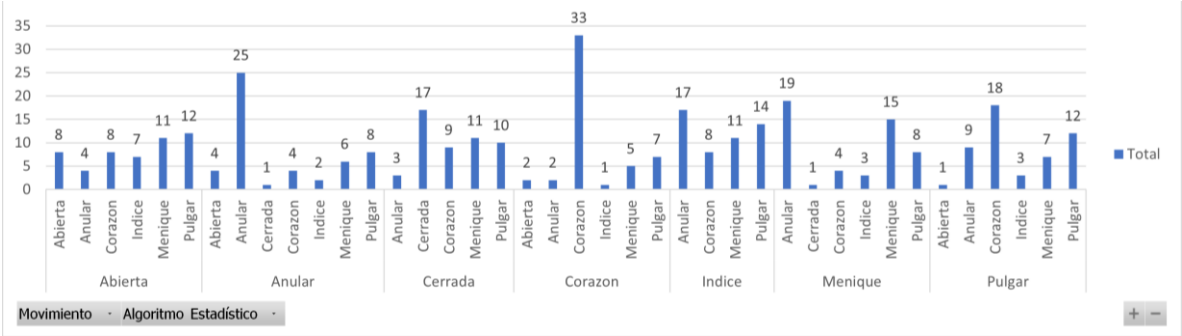

Figura 31. Resultados Algoritmo Estadístico. Tomado de: Autor.

Estos resultados confirman que las redes neuronales obtienen mejores resultados al clasificar señales EMG que los algoritmos estadísticos.

MATRIZ DE CONFUSIÓN								
ALGORITMO	**MOVIMIENTO**	**TP**	**FN**	**TN**	**FP**	**Sensitivity (%)**	**Specificity (%)**	**Accuracy (%)**
RED NEURONAL	Meñique	32	18	275	25	64	91.66666667	87.71428571
	Anular	26	24	282	18	52	94	88
	Corazón	39	11	294	6	78	98	95.14285714
	Indice	34	16	270	30	68	90	86.85714286
	Pulgar	48	2	266	34	96	88.66666667	89.71428571
	Abierta	26	24	296	4	52	98.66666667	92
	Cerrada	27	23	299	1	54	99.66666667	93.14285714
ALGORITMO ESTADÍSTICO	Meñique	15	35	249	51	30	83	75.42857143
	Anular	25	25	246	54	50	82	77.42857143
	Corazón	33	17	249	51	66	83	80.57142857
	Indice	0	50	284	16	0	94.66666667	81.14285714
	Pulgar	12	38	241	59	24	80.33333333	72.28571429
	Abierta	8	42	293	7	16	97.66666667	86
	Cerrada	17	33	298	2	34	99.33333333	90

Tabla 6. Matriz de confusión: Redes Neuronales vs. Algoritmos estadísticos. Tomado de: Autor.

8. CONCLUSIONES Y TRABAJO FUTURO

8.1 GENERAL

- El algoritmo de reconocimiento de movimientos de la mano, diseñado con una red neuronal artificial, identifica correctamente el movimiento realizado por el usuario el 75.71% de las veces, sin embargo, presenta dificultad para reconocer el movimiento del dedo anular puesto que es un dedo que comparte su estructura muscular con los dedos meñique y corazón además de tener poco grado de libertad.

8.2 ESPECÍFICOS

- El estudio previo del origen de las señales electromiográficas permitió concluir que la similitud encontrada en las señales de los dedos meñique, anular y corazón se deben a la conexión muscular de los mismos. También permitió comprender que, si la frecuencia natural de dos dedos es similar, un algoritmo de conteo de cruces por cero se ve limitado para identificar el movimiento realizado.
- Se pudo observar durante las pruebas del algoritmo con la red neuronal artificial, que la exactitud del reconocimiento es directamente proporcional a la cantidad de muestras con la cual se entrena, es decir, cuantos más ejemplos tenga el algoritmo le será más sencillo identificar las características del movimiento de cada dedo y disminuir el error durante el entrenamiento.
- Un algoritmo de aprendizaje automático tiene la capacidad de reajustarse y encontrar similitudes entre un grupo de señales, sin embargo, tiene como limitación la posibilidad del sobre entrenamiento, donde el algoritmo empieza a aprender la forma de las señales y puede incrementar el error al evaluar señales que no conozca.
- El trabajo realizado muestra que los algoritmos de aprendizaje automático obtienen mejores resultados al procesar señales biológicas que los métodos estadísticos, esto debido a que una red neuronal tiene la capacidad de ajustarse para encontrar patrones y así identificar similitudes en señales con las que no haya sido entrenado, mientras que un algoritmo estadístico se ve limitado a la tendencia que presente un conjunto de datos.

8.3 TRABAJO FUTURO

Para futuros desarrollos o mejoras a este trabajo se sugiere incrementar la base de datos ya que entre más muestras tenga le será más sencillo identificar las señales y aumentará el porcentaje de aciertos. También se podría implementar una interfaz que permita la identificación en tiempo real de los movimientos ya que esto haría

posible la integración del desarrollo con una prótesis de brazo como la que se encuentra en la Facultad de Ingeniería Mecánica de la Universidad Santo Tomás. Ademas se planea implemantar un algoritmo de aprendizaje profundo basado en una arquitectura “Long Short Term Memory” (LSTM) para la deteccion de los movimientos de la mano y poder compararlo con los resultados obtenidos previamente. Dicho trabajo se realizará de forma similar al previamente desarrollado en [17].

REFERENCIAS BIBLIOGRÁFICAS

[1] J. Kimura, Electrodiagnosis in Diseases of Nerve and Muscle : Principles and Practice, New York: Oxford University Press, 2013.

[2] J. A. L. S. Eduardo Francisco Caicedo Bravo, Una aproximación práctica a las redes neuronales artificiales, Cali, Colombia: Programa Editorial Universidad del Valle, 2009.

[3] C. F. Valencia, J. A. Suarez, A. Cogollos, R. A. Uribe y G. C. Flores, «Heridos en combate, experiencia del Grupo de Trauma del Hospital Militar Central de Bogotá,» *Revista Colombiana de Cirugía,* vol. 30, nº 1, pp. 18-23, 2015.

[4] Z.-G. Kathryn, J. Ellen, L. Patti, G. Thomas y B. Ron, «Estimating the Prevalence of Limb Loss in the United States: 2005 to 2050,» *Archives of Physical Medicine and Rehabilitation,* vol. 89, nº 3, pp. 422-429, 2008.

[5] C. D. Aguila, Electromedicina, 2 ed., Buenos Aires: Presencia Ltda, 1994, pp. 225-238.

[6] J. Moore y G. Zouridakis, Biomedical Technology and Devices Handbook, United States of America: CRC Press, 2004.

[7] L. Bi, C. Guan y G. Feleke, «A review on EMG-based motor intention prediction of continuous human upper limb motion for human-robot collaboration,» *Biomedical Signal Processing and Control,* vol. 51, pp. 113-127, 2018.

[8] J. Mads, W. Asim y K. Ernest Nlandu, «The effect of arm position on classification of hand gestures with intramuscular EMG,» *Biomedical Signal Processing and Control,* vol. 43, pp. 1-8, 2018.

[9] N. Emiliano, B. Alberto Dellacasa, C. Anna Lisa, S. Rinaldo, D. Angelo, G. Eugenio y Z. Loredana, «EMG and ENG-envelope pattern recognition for prosthetic hand control,» *Journal of Neuroscience Methods,* vol. 311, pp. 38-46, 2019.

[10] D. B. Alberto, G. Emanuele, C. Giorgio, D. Angelo, S. Rinaldo, G. Eugenio y Z. Loredana, «NLR, MLP, SVM, and LDA: a comparative analysis on EMG data from people with trans-radial amputation,» *Journal of NeuroEngineering and Rehabilitation,* 2017.

[11] A. Ciancio, R. Barone, L. Zollo, G. Carpino, A. Davalli, R. Sacchetti y E. Guglielmelli, «A bio-inspired force control for cyclic manipulation of prosthetic hands,» *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC),* pp. 4824-4827, 2015.

[12] N. Chaiyaratana, Zalzala y D. Datta, «Myoelectric Signals Pattern Recognition for Intelligent Functional Operation of Upper-Limb Prosthesis,» *White Rose,* nº 621, 1996.

[13] B. Roberto, A. L. Ciancio, R. A. Romeo, A. Davall, R. S. E. Guglielmelli y L. Zollo, «Multilevel control of an anthropomorphic prosthetic hand for grasp and slip prevention,» *Advances in Mechanical Engineering,* vol. 8, nº 9, 2016.

[14] J. P. T. Portillo, Introducción a las señales y sistemas, Barranquilla: Universidad del Norte, 2017.

[15] L. Tan, Digital Signal Processing : Fundamentals and Applications, Amsterdam: Academic Press, 2008.

[16] N. Y. M. F. Christian Kanzow, «Levenberg–Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints,» *Journal of Computational and Applied Mathematics,* vol. 172, nº 2, pp. 375-397, 2004.

[17] A. Reyes, E. Camacho, A. Mateus y J. M. Calderón, «LSTM based brain-machine interface tool for text generation through eyes blinking detection,» *IEEE 18th Annual Consumer Communications & Networking Conference (CCNC),* pp. 1-6, 2021.

[18] NORTH, «Downloads Myo Support,» [En línea]. Available: https://support.getmyo.com/hc/en-us/articles/360018409792-Myo-Connect-SDK-and-firmware-downloads.

[19] B. M. Kte'pi, «MATLAB (matrix laboratory).,» *Salem Press Encyclopedia of Science,* p. 1, 2020.

[20] M. B. Velasco, F. C. Roldán, R. J. Martínez y J. S. Landete, Tratamiento Digital de Señales, Universidad de Alcalá, 2013.

[21] E. Ogier, Multi-layer perceptron, MATLAB Central File Exchange, 2018.

ANEXOS

1. Código de Muestreo y Recorte de Señales

```
clc
clear all
close all

%% INSTALACION DE MYO_MEX
install_myo_mex;
sdk_path = 'C:\myo-sdk-win-0.9.0';
build_myo_mex(sdk_path);

% RECIBIR DATOS DE CONEXION DEL DISPOSITIVO
TIME_DURATION = 20; % seconds
countMyos = 1;

fprintf('Collecting data from %d Myos for roughly %5.2f[s] ... ',...
  countMyos,TIME_DURATION);
mm = MyoMex(countMyos);
m = mm.myoData;
T = 5; % seconds
m.clearLogs()
m.startStreaming();
pause(T);
m.stopStreaming();

%% PROCESO DE RECORTE DE SENALES
signal = m.emg_log(:,3)
% BUSCAR LOS PICOS
[pks,locs] = findpeaks(signal,'MinPeakDistance' , 8)

% UBICACION DEL PICO MAXIMO
M=max(pks);
for N=1:length(pks)
    if M == pks(N)
        M = N;
    end
end

% DEFINIR LA VENTANA SEGUN EL PICO MAXIMO
% Se toman 150 datos con el punto maximo en medio.
snew=signal((locs(M))-74:(locs(M))+75);

% PINTAR LA SEÑAL ORIGINAL Y LA RECORTADA
hold on
subplot(2,1,1),plot(signal), title('Senal Original'),xlabel('Dato de la
muestra'),ylabel('Potencia (W)')
subplot(2,1,2),plot(snew), title('Senal
Recortada'),xlabel('Dato'),ylabel('Potencial Electrico (w)')
hold off
```

```
%% DESTRUIR LA CONEXION
mm.delete();
fprintf('Done!\n\n');
```

2. Código para Adecuación de la señal

```
clc
clear all
close all
nMuestras = input('Ingrese número de muestras: ');
tic
for electrodo = 1:3
    electrodox = int2str(electrodo)

    for muestra = 1:nMuestras
%         mx=int2str(muestra);
        contador = muestra+10;
        if (contador >= nMuestras)
            contador = nMuestras;
        end
        MD1=0;
        AD1=0;
        CD1=0;
        ID1=0;
        PD1=0;
        ABD1=0;
        CED1=0;
        for item=muestra:contador
            if(item ~= 0)
                mx=int2str(item);
                MD1=
MD1+importdata(strcat('Dataset\Menique\SE',electrodox,'_',mx,'.mat'));
                AD1=
AD1+importdata(strcat('Dataset\Anular\SE',electrodox,'_',mx,'.mat'));
                CD1=
CD1+importdata(strcat('Dataset\Corazon\SE',electrodox,'_',mx,'.mat'));
                ID1=
ID1+importdata(strcat('Dataset\Indice\SE',electrodox,'_',mx,'.mat'));
                PD1=
PD1+importdata(strcat('Dataset\Pulgar\SE',electrodox,'_',mx,'.mat'));
                ABD1=
ABD1+importdata(strcat('Dataset\Abierta\SE',electrodox,'_',mx,'.mat'));
                CED1=
CED1+importdata(strcat('Dataset\Cerrada\SE',electrodox,'_',mx,'.mat'));
            end
        end
%         MD1=
importdata(strcat('Dataset\Menique\SE',electrodox,'_',mx,'.mat'));
%         AD1=
importdata(strcat('Dataset\Anular\SE',electrodox,'_',mx,'.mat'));
%         CD1=
importdata(strcat('Dataset\Corazon\SE',electrodox,'_',mx,'.mat'));
%         ID1=
importdata(strcat('Dataset\Indice\SE',electrodox,'_',mx,'.mat'));
```

```
%          PD1=
importdata(strcat('Dataset\Pulgar\SE',electrodox,'_',mx,'.mat'));

        % SENAL DE ENTRADA
        x=MD1+AD1+CD1+ID1+PD1+ABD1+CED1;

        % DATOS DE RETROALIMENTACIÓN
        N=46; %16Resolucion
        R=4; %4
        XR=zeros(R);
        % DATOS DE ENTRENAMIENTO
        Alfa=0.0001;
        X(1:N)=0;
        X(N+1:N+R)=0;
        Y=0;
        % CARGAR PESOS
        if muestra == 1
            WM=(randn(1,N+R))/100;
            WA=(randn(1,N+R))/100;
            WC=(randn(1,N+R))/100;
            WI=(randn(1,N+R))/100;
            WP=(randn(1,N+R))/100;
            WAB=(randn(1,N+R))/100;
            WCE=(randn(1,N+R))/100;

save(strcat('Pesos_',electrodox,'.mat'),'WM','WA','WC','WI','WP','WAB','W
CE');
        else
            load(strcat('Pesos_',electrodox,'.mat'));
        end

        % SENAL DE SALIDA
        for dedo=1:7
            % CLASIFICACION POR DEDOS
            switch dedo
                case 1
                    y = MD1;
                    W = WM;
                case 2
                    y = AD1;
                    W = WA;
                case 3
                    y = CD1;
                    W = WC;
                case 4
                    y = ID1;
                    W = WI;
                case 5
                    y = PD1;
                    W = WP;
                case 6
                    y = ABD1;
                    W = WAB;
                case 7
```

```
        y = CED1;
        W = WCE;
end

% ENTRENAMIENTO
for epocas=1:2000

    % CALCULO DE PESOS
    for p=1:150
        %DESPLAZAMIENTO DEL VECTOR
        X(2:N)=X(1:N-1);
        X(1)=x(p);
        X(N+2:N+R)=X(N+1:N+R-1);
        X(N+1)=Y;
        %SIMULACION DE Y
        Y=sum(X.*W);
        % CALCULO DE ERROR
        Error=y(p)-Y;
        % AJUSTE DE PESOS
        W=W+Alfa*Error*X;
    end

    % ENTRENAMIENTO
    Ys(1)=0;
    for p=2:150
        X(2:N)=X(1:N-1);
        X(1)=x(p);
        X(N+2:N+R)=X(N+1:N+R-1);
        X(N+1)=Ys(p-1);
        % CALCULO DE LA SALIDA
        Ys(p)=sum(X.*W);
    end

    % CALCULO DEL ERROR
    ErrorH(epocas)=(sum(abs(Ys-y')))/100;

end

% VERIFICACION DE VARIABLES
if (isnan(W) | isequal(W,0))
    W
    dedo
    electrodo
    muestra
    pause
end

% GURDADO DE PESOS
switch dedo
    case 1
        WM = W;
    case 2
        WA = W;
    case 3
```

```
                    WC = W;
                case 4
                    WI = W;
                case 5
                    WP = W;
                case 6
                    WAB = W;
                case 7
                    WCE = W;
            end

save(strcat('Pesos_',electrodox,'.mat'),'WM','WA','WC','WI','WP','WAB','W
CE');
        end
    end
%     % IMPRIMIR ERROR POR CICLO
%     figure('Name','Error','NumberTitle','off')
%     plot(ErrorH),title('Error')
%     % IMPRIMIR SALIDA
%     figure('Name','Salida','NumberTitle','off')
%     plot(1:150,Ys,'r',1:150,y,'b'),title('Salida')
%     legend('Salida Real','Salida Deseada')
end
toc
% IMPRIMIR ERROR
figure('Name','Error','NumberTitle','off')
plot(ErrorH),title('Error durante el
Entrenamiento'),xlabel('Epoca'),ylabel('Error')
% IMPRIMIR SALIDA
figure('Name','Salida','NumberTitle','off')
plot(1:150,Ys,'r',1:150,y,'b'),title('Salida'),xlabel('Datos'),ylabel('Po
tencia (W)')
legend('Salida Real','Salida Deseada')
```

3. Código para el Filtrado de la señal

```
clc
clear all
nMuestras = input('Ingrese número de muestras: ');
for electrodo=1:3
    electrodox=int2str(electrodo);
    load(strcat('Pesos_',electrodox,'.mat'));

    for muestra=1:nMuestras

        muestrax=int2str(muestra);
        % DATOS DE RETROALIMENTACIÓN
        N=46; %Resolucion
        R=4;
        XR=zeros(R);

        for dedo=1:7

            % CLASIFICACION POR DEDOS
```

```
            switch dedo
                case 1
                    v='Menique.mat';
                    D1=
importdata(strcat('Dataset\Menique\SE',electrodox,'_',muestrax,'.mat'));
                case 2
                    v='Anular.mat';
                    D1=
importdata(strcat('Dataset\Anular\SE',electrodox,'_',muestrax,'.mat'));
                case 3
                    v='Corazon.mat';
                    D1=
importdata(strcat('Dataset\Corazon\SE',electrodox,'_',muestrax,'.mat'));
                case 4
                    v='Indice.mat';
                    D1=
importdata(strcat('Dataset\Indice\SE',electrodox,'_',muestrax,'.mat'));
                case 5
                    v='Pulgar.mat';
                    D1=
importdata(strcat('Dataset\Pulgar\SE',electrodox,'_',muestrax,'.mat'));
                case 6
                    v='Abierta.mat';
                    D1=
importdata(strcat('Dataset\Abierta\SE',electrodox,'_',muestrax,'.mat'));
                case 7
                    v='Cerrada.mat';
                    D1=
importdata(strcat('Dataset\Cerrada\SE',electrodox,'_',muestrax,'.mat'));
            end

            % FILTRADO (FILTRO IRR)

            % MENIQUE
            X(1:N)=0;
            X(N+1:N+R)=0;
            YM(1)=0;
            for p=2:150

                X(2:N)=X(1:N-1);
                X(1)=D1(p);
                X(N+2:N+R)=X(N+1:N+R-1);
                X(N+1)=YM(p-1);

                YM(p)=sum(X.*WM);

            end
            PM= (sum(abs(YM)))/(sum(abs(D1)));

            % PULGAR
            X(1:N)=0;
            X(N+1:N+R)=0;
            YP(1)=0;
            for p=2:150
```

```
    X(2:N)=X(1:N-1);
    X(1)=D1(p);
    X(N+2:N+R)=X(N+1:N+R-1);
    X(N+1)=YP(p-1);

    YP(p)=sum(X.*WP);

end
PP= (sum(abs(YP)))/(sum(abs(D1)));

% ANULAR
X(1:N)=0;
X(N+1:N+R)=0;
YA(1)=0;
for p=2:150

    X(2:N)=X(1:N-1);
    X(1)=D1(p);
    X(N+2:N+R)=X(N+1:N+R-1);
    X(N+1)=YA(p-1);

    YA(p)=sum(X.*WA);

end
PA= (sum(abs(YA)))/(sum(abs(D1)));

% CORAZON
X(1:N)=0;
X(N+1:N+R)=0;
YC(1)=0;
for p=2:150

    X(2:N)=X(1:N-1);
    X(1)=D1(p);
    X(N+2:N+R)=X(N+1:N+R-1);
    X(N+1)=YC(p-1);

    YC(p)=sum(X.*WC);

end
PC= (sum(abs(YC)))/(sum(abs(D1)));

% INDICE
X(1:N)=0;
X(N+1:N+R)=0;
YI(1)=0;
for p=2:150

    X(2:N)=X(1:N-1);
    X(1)=D1(p);
    X(N+2:N+R)=X(N+1:N+R-1);
    X(N+1)=YI(p-1);
```

```
            YI(p)=sum(X.*WI);

        end
        PI= (sum(abs(YI)))/(sum(abs(D1)));

        % ABIERTA
        X(1:N)=0;
        X(N+1:N+R)=0;
        YAB(1)=0;
        for p=2:150

            X(2:N)=X(1:N-1);
            X(1)=D1(p);
            X(N+2:N+R)=X(N+1:N+R-1);
            X(N+1)=YAB(p-1);

            YAB(p)=sum(X.*WAB);

        end
        PAB= (sum(abs(YAB)))/(sum(abs(D1)));

        % CERRADA
        X(1:N)=0;
        X(N+1:N+R)=0;
        YCE(1)=0;
        for p=2:150

            X(2:N)=X(1:N-1);
            X(1)=D1(p);
            X(N+2:N+R)=X(N+1:N+R-1);
            X(N+1)=YCE(p-1);

            YCE(p)=sum(X.*WCE);

        end
        PCE= (sum(abs(YCE)))/(sum(abs(D1)));

        if exist(v)== 0
            datos=[PM,PA,PC,PI,PP,PAB,PCE];
            save(v,'datos')
        else
            tabla=[PM,PA,PC,PI,PP,PAB,PCE];
            load (v)
            datos=[datos;tabla];
            save(v,'datos')
        end

    end
end
mkdir(strcat('Electrodo_',electrodox));
movefile('Menique.mat',strcat('Electrodo_',electrodox));
movefile('Anular.mat',strcat('Electrodo_',electrodox));
movefile('Corazon.mat',strcat('Electrodo_',electrodox));
movefile('Indice.mat',strcat('Electrodo_',electrodox));
```

```
    movefile('Pulgar.mat',strcat('Electrodo_',electrodox));
    movefile('Abierta.mat',strcat('Electrodo_',electrodox));
    movefile('Cerrada.mat',strcat('Electrodo_',electrodox));
end
```

4. Entrenamiento de la Red Neuronal Artificial

```
clc
clear all
close all
Entradas=[];
Salida=[];
nMuestras = input('Ingrese número de muestras: ');
for Ejemplos = 1:nMuestras
    for nDedo = 1:7
        switch nDedo
            case 1
                dedo = 'Menique';
                Salida = [Salida [1;0;0;0;0;0;0]];
            case 2
                dedo = 'Anular';
                Salida = [Salida [0;1;0;0;0;0;0]];
            case 3
                dedo = 'Corazon';
                Salida = [Salida [0;0;1;0;0;0;0]];
            case 4
                dedo = 'Indice';
                Salida = [Salida [0;0;0;1;0;0;0]];
            case 5
                dedo = 'Pulgar';
                Salida = [Salida [0;0;0;0;1;0;0]];
            case 6
                dedo = 'Abierta';
                Salida = [Salida [1;0;0;0;0;1;0]];
            case 7
                dedo = 'Cerrada';
                Salida = [Salida [1;0;0;0;0;0;1]];
        end
        Aux=[];
        for electrodo=1:3
            selectrodo=int2str(electrodo);

datos=importdata(strcat('Electrodo_',selectrodo,'\',dedo,'.mat'));
            Aux = [Aux,datos(Ejemplos,:)];
        end
        Entradas = [Entradas, Aux'];
    end
end

MLP = ...
    MultiLayerPerceptron('LengthsOfLayers', [21 7 7],...
                         'HiddenActFcn',     'tanh',...
                         'OutputActFcn',     'softmax');
```

```
% Options
Options = ...
    struct('TrainingAlgorithm', 'LM',...
           'NumberOfEpochs',    100,...
           'MinimumMSE',        1e-3,...
           'InitialMarquardtParameter', 0.01,...
           'MaximumMarquardtParameter', 1e10,...
           'MarquardtFactor',           10,...
           'BayesianRegularization',    false);
% Options = ...
%     struct('TrainingAlgorithm', 'GD',...
%            'NumberOfEpochs',    5000,...
%            'MinimumMSE',        1e-3,...
%            'SizeOfBatches',     40,...
%            'SplitRatio',        0.7,...
%            'Momentum',          0.9);

% Training
MLP.train(Entradas,Salida,Options);

% Save Outputs
save('MLP.mat','MLP')
```

5. Conteo de Cruces por Cero

```
clc
clear all
close all
nMuestras = input('Ingrese número de muestras: ');

for electrodo = 1:3
    electrodox = int2str(electrodo);
    M(electrodo) = 0;
    A(electrodo) = 0;
    C(electrodo) = 0;
    I(electrodo) = 0;
    P(electrodo) = 0;
    AB(electrodo) = 0;
    CE(electrodo) = 0;
    for muestra = 1:nMuestras
        mx=int2str(muestra);

MD1=importdata(strcat('Dataset\Menique\SE',electrodox,'_',mx,'.mat'));
        AD1=
importdata(strcat('Dataset\Anular\SE',electrodox,'_',mx,'.mat'));
        CD1=
importdata(strcat('Dataset\Corazon\SE',electrodox,'_',mx,'.mat'));
        ID1=
importdata(strcat('Dataset\Indice\SE',electrodox,'_',mx,'.mat'));
        PD1=
importdata(strcat('Dataset\Pulgar\SE',electrodox,'_',mx,'.mat'));
        ABD1=
importdata(strcat('Dataset\Abierta\SE',electrodox,'_',mx,'.mat'));
```

```
        CED1=
importdata(strcat('Dataset\Cerrada\SE',electrodox,'_',mx,'.mat'));

        % MENIQUE
        ceros=[];
        for i=1:(length(MD1)-1)
            if (MD1(i)*MD1(i+1))==0
                if MD1(i)==0
                    ceros=[ceros;i];
                else
                    ceros=[ceros;i+1];
                end
            elseif (MD1(i)*MD1(i+1))<0
                if abs(MD1(i))<abs(MD1(i+1))
                    ceros=[ceros;i];
                else
                    ceros=[ceros;i+1];
                end
            end
        end
        M(electrodo) = M(electrodo)+length(ceros);

        % ANULAR
        ceros=[];
        for i=1:(length(AD1)-1)
            if (AD1(i)*AD1(i+1))==0
                if AD1(i)==0
                    ceros=[ceros;i];
                else
                    ceros=[ceros;i+1];
                end
            elseif (AD1(i)*AD1(i+1))<0
                if abs(AD1(i))<abs(AD1(i+1))
                    ceros=[ceros;i];
                else
                    ceros=[ceros;i+1];
                end
            end
        end
        A(electrodo) = A(electrodo)+length(ceros);

        % CORAZON
        ceros=[];
        for i=1:(length(CD1)-1)
            if (CD1(i)*CD1(i+1))==0
                if CD1(i)==0
                    ceros=[ceros;i];
                else
                    ceros=[ceros;i+1];
                end
            elseif (CD1(i)*CD1(i+1))<0
                if abs(CD1(i))<abs(CD1(i+1))
                    ceros=[ceros;i];
                else
```

```
            ceros=[ceros;i+1];
        end
    end
end
C(electrodo) = C(electrodo)+length(ceros);

% INDICE
ceros=[];
for i=1:(length(ID1)-1)
    if (ID1(i)*ID1(i+1))==0
        if ID1(i)==0
            ceros=[ceros;i];
        else
            ceros=[ceros;i+1];
        end
    elseif (ID1(i)*ID1(i+1))<0
        if abs(ID1(i))<abs(ID1(i+1))
            ceros=[ceros;i];
        else
            ceros=[ceros;i+1];
        end
    end
end
I(electrodo) = I(electrodo)+length(ceros);

% PULGAR
ceros=[];
for i=1:(length(PD1)-1)
    if (PD1(i)*PD1(i+1))==0
        if PD1(i)==0
            ceros=[ceros;i];
        else
            ceros=[ceros;i+1];
        end
    elseif (PD1(i)*PD1(i+1))<0
        if abs(PD1(i))<abs(PD1(i+1))
            ceros=[ceros;i];
        else
            ceros=[ceros;i+1];
        end
    end
end
P(electrodo) = P(electrodo)+length(ceros);

% ABIERTA
ceros=[];
for i=1:(length(ABD1)-1)
    if (ABD1(i)*ABD1(i+1))==0
        if ABD1(i)==0
            ceros=[ceros;i];
        else
            ceros=[ceros;i+1];
        end
    elseif (ABD1(i)*ABD1(i+1))<0
```

```
                if abs(ABD1(i))<abs(ABD1(i+1))
                    ceros=[ceros;i];
                else
                    ceros=[ceros;i+1];
                end
            end
        end
        AB(electrodo) = AB(electrodo)+length(ceros);

        % CERRADA
        ceros=[];
        for i=1:(length(CED1)-1)
            if (CED1(i)*CED1(i+1))==0
                if CED1(i)==0
                    ceros=[ceros;i];
                else
                    ceros=[ceros;i+1];
                end
            elseif (CED1(i)*CED1(i+1))<0
                if abs(CED1(i))<abs(CED1(i+1))
                    ceros=[ceros;i];
                else
                    ceros=[ceros;i+1];
                end
            end
        end
        CE(electrodo) = CE(electrodo)+length(ceros);
    end
    M(electrodo) = M(electrodo)/nMuestras;
    A(electrodo) = A(electrodo)/nMuestras;
    C(electrodo) = C(electrodo)/nMuestras;
    I(electrodo) = I(electrodo)/nMuestras;
    P(electrodo) = P(electrodo)/nMuestras;
    AB(electrodo) = AB(electrodo)/nMuestras;
    CE(electrodo) = CE(electrodo)/nMuestras;
end
```

6. Interfaz Gráfica en App Designer

```
classdef AppMyo < matlab.apps.AppBase


    % Properties that correspond to app components
    properties (Access = public)
        UIFigure             matlab.ui.Figure
        GrabarButton         matlab.ui.control.Button
        Image                matlab.ui.control.Image
        Titulo               matlab.ui.control.Label
        textoEstado          matlab.ui.control.Label
        DedosButtonGroup     matlab.ui.container.ButtonGroup
        MeniqueButton        matlab.ui.control.RadioButton
```

```
    AnularButton          matlab.ui.control.RadioButton
    CorazonButton         matlab.ui.control.RadioButton
    IndiceButton          matlab.ui.control.RadioButton
    PulgarButton          matlab.ui.control.RadioButton
    AbiertaButton         matlab.ui.control.RadioButton
    CerradaButton         matlab.ui.control.RadioButton
    GuardarButton         matlab.ui.control.Button
    MuestraSpinnerLabel   matlab.ui.control.Label
    MuestraSpinner        matlab.ui.control.Spinner
    EntrenarButton        matlab.ui.control.Button
    textoComparacion      matlab.ui.control.Label
    Grafica               matlab.ui.control.UIAxes
end



properties (Access = public)
    datosE1 = [];
    datosE2 = [];
    datosE3 = [];
    datosF = [];
    MLP;
    nmuestra = 0;
end



% Callbacks that handle component events
methods (Access = private)


    % Button pushed function: GrabarButton
    function GrabarButtonPushed(app, event)
        app.nmuestra = app.MuestraSpinner.Value;
        app.GrabarButton.BackgroundColor = 'r';
        % INSTALACION DE MYO_MEX
        app.textoEstado.Text = 'Iniciando MyoMex...';
        app.textoComparacion.Text = '';
        drawnow
        install_myo_mex;
        sdk_path = 'C:\myo-sdk-win-0.9.0';
        build_myo_mex(sdk_path);

        % RECIBIR DATOS DE CONEXION DEL DISPOSITIVO
        app.textoEstado.Text = 'Conectando con el sensor...';
        drawnow
        countMyos = 1;
```

```
            m1 = MyoMex(countMyos);
            m = m1.myoData;
            pause(1);
            m.clearLogs()
            app.GrabarButton.BackgroundColor = 'g';
            m.startStreaming();
            app.textoEstado.Text = 'Recibiendo datos...';
            drawnow
            pause(3);
            m.stopStreaming();
            app.GrabarButton.BackgroundColor = 'w';
            app.textoEstado.Text = 'Hecho!';
            drawnow

% BUSCAR LOS PICOS
signalE1 = m.emg_log(:,1);
signalE2 = m.emg_log(:,2);
signalE3 = m.emg_log(:,3);
[pks,locs] = findpeaks(signalE1,'MinPeakDistance' , 8);
% UBICACION DEL PICO MAXIMO
M=max(pks);
for N=1:length(pks)
if M == pks(N)
M = N;
end
end
% FINALIZAR LA CONEXION
m1.delete();
% DEFINIR LA VENTANA SEGUN EL PICO MAXIMO
% Se toman 150 datos con el punto maximo en medio.
if (locs(M)+75) > length(signalE1) || (locs(M)-74) <= 0
app.textoEstado.Text = 'Senal incompleta. Intentelo nuevamente.';
drawnow
else
snewE1=signalE1((locs(M))-74:(locs(M))+75);
snewE2=signalE2((locs(M))-74:(locs(M))+75);
snewE3=signalE3((locs(M))-74:(locs(M))+75);
% DATOS PARA GUARDAR
app.datosE1 = snewE1;
app.datosE2 = snewE2;
app.datosE3 = snewE3;
% PINTAR LA SEÑAL
cla(app.Grafica)
hold(app.Grafica);
plot(app.Grafica,signalE1)
% FILTRADO
app.datosF = [];
for electrodo=1:3
```

```
electrodox=int2str(electrodo);
load(strcat('Pesos_',electrodox,'.mat'), 'WA', 'WC', 'WM', 'WI',
'WP','WAB','WCE');
% DATOS DE RETROALIMENTACIÓN
N=46; %Resolucion
R=4;
% FILTROS (FILTRO IRR)
% CARGA DATOS PARA FILTRADO
switch (electrodo)
case 1
muestra = app.datosE1;
case 2
muestra = app.datosE2;
case 3
muestra = app.datosE3;
end
% MENIQUE
X(1:N)=0;
X(N+1:N+R)=0;
YM(1)=0;
for p=2:150
X(2:N)=X(1:N-1);
X(1)=muestra(p);
X(N+2:N+R)=X(N+1:N+R-1);
X(N+1)=YM(p-1);
YM(p)=sum(X.*WM);
end
PM= (sum(abs(YM)))/(sum(abs(muestra)));
% PULGAR
X(1:N)=0;
X(N+1:N+R)=0;
YP(1)=0;
for p=2:150
X(2:N)=X(1:N-1);
X(1)=muestra(p);
X(N+2:N+R)=X(N+1:N+R-1);
X(N+1)=YP(p-1);
YP(p)=sum(X.*WP);
end
PP= (sum(abs(YP)))/(sum(abs(muestra)));
% ANULAR
X(1:N)=0;
X(N+1:N+R)=0;
YA(1)=0;
for p=2:150
X(2:N)=X(1:N-1);
X(1)=muestra(p);
X(N+2:N+R)=X(N+1:N+R-1);
```

```
X(N+1)=YA(p-1);
YA(p)=sum(X.*WA);
end
PA= (sum(abs(YA)))/(sum(abs(muestra)));
% CORAZON
X(1:N)=0;
X(N+1:N+R)=0;
YC(1)=0;
for p=2:150
X(2:N)=X(1:N-1);
X(1)=muestra(p);
X(N+2:N+R)=X(N+1:N+R-1);
X(N+1)=YC(p-1);
YC(p)=sum(X.*WC);
end
PC= (sum(abs(YC)))/(sum(abs(muestra)));
% INDICE
X(1:N)=0;
X(N+1:N+R)=0;
YI(1)=0;
for p=2:150
X(2:N)=X(1:N-1);
X(1)=muestra(p);
X(N+2:N+R)=X(N+1:N+R-1);
X(N+1)=YI(p-1);
YI(p)=sum(X.*WI);
end
PI= (sum(abs(YI)))/(sum(abs(muestra)));
% ABIERTA
X(1:N)=0;
X(N+1:N+R)=0;
YAB(1)=0;
for p=2:150
X(2:N)=X(1:N-1);
X(1)=muestra(p);
X(N+2:N+R)=X(N+1:N+R-1);
X(N+1)=YAB(p-1);
YAB(p)=sum(X.*WAB);
end
PAB= (sum(abs(YAB)))/(sum(abs(muestra)));
% CERRADA
X(1:N)=0;
X(N+1:N+R)=0;
YCE(1)=0;
for p=2:150
X(2:N)=X(1:N-1);
X(1)=muestra(p);
X(N+2:N+R)=X(N+1:N+R-1);
```

```
X(N+1)=YI(p-1);
YCE(p)=sum(X.*WCE);
end
PCE= (sum(abs(YCE)))/(sum(abs(muestra)));
tabla=[PM,PA,PC,PI,PP,PAB,PCE];
app.datosF=[app.datosF, tabla]
end
app.datosF
% SALIDA
app.MLP.propagate(app.datosF');
Probabilities = app.MLP.Outputs;
[X, Prob] = max(Probabilities,[],1);
Prob
% RESPUESTA RED NEURONAL
switch (Prob)
case 1
app.textoEstado.Text = 'Red Neuronal: Menique';
app.Image.ImageSource = 'Menique.jpg';
case 2
app.textoEstado.Text = 'Red Neuronal: Anular';
app.Image.ImageSource = 'Anular.jpg';
case 3
app.textoEstado.Text = 'Red Neuronal: Corazon';
app.Image.ImageSource = 'Corazon.jpg';
case 4
app.textoEstado.Text = 'Red Neuronal: Indice';
app.Image.ImageSource = 'Indice.jpg';
case 5
app.textoEstado.Text = 'Red Neuronal: Pulgar';
app.Image.ImageSource = 'Pulgar.jpg';
case 6
app.textoEstado.Text = 'Red Neuronal: Abierta';
app.Image.ImageSource = 'Abierta.png';
case 7
app.textoEstado.Text = 'Red Neuronal: Cerrada';
app.Image.ImageSource = 'Cerrada.png';
end
% ALGORITMO ESTADISTICO
% CRUCES POR CERO
MD(:,1) = app.datosE1;
MD(:,2) = app.datosE2;
MD(:,3) = app.datosE3;
for electrodo=1:3
M(electrodo) = 0;
ceros=[];
MD1 = MD(:,electrodo);
for i=1:(length(MD1)-1)
if (MD1(i)*MD1(i+1))==0
```

```
if MD1(i)==0
ceros=[ceros;i];
else
ceros=[ceros;i+1];
end
elseif (MD1(i)*MD1(i+1))<0
if abs(MD1(i))<abs(MD1(i+1))
ceros=[ceros;i];
else
ceros=[ceros;i+1];
end
end
end
M(electrodo) = M(electrodo)+length(ceros);
end
M
% PROMEDIO DE CRUCES POR CERO
MMe=[95.4700, 92.9367, 90.5933];
AMe=[98.0867, 90.3300, 88.4567];
CMe=[97.4300, 95.2033, 97.6067];
IMe=[97.1467, 92.4267, 90.4067];
PMe=[98.3767, 95.8167, 95.2600];
ABMe=[95.2433, 96.9267, 97.7900];
CEMe=[94.9000, 92.7900, 93.6067];
% DISTANCIA EUCLIDIANA
% MENIQUE
Dis(1) = sqrt(((MMe(1)-M(1))^2)+((MMe(2)-M(2))^2)+((MMe(3)-M(3))^2));
% ANULAR
Dis(2) = sqrt(((AMe(1)-M(1))^2)+((AMe(2)-M(2))^2)+((AMe(3)-M(3))^2));
% CORAZON
Dis(3) = sqrt(((CMe(1)-M(1))^2)+((CMe(2)-M(2))^2)+((CMe(3)-M(3))^2));
% INDICE
Dis(4) = sqrt(((IMe(1)-M(1))^2)+((IMe(2)-M(2))^2)+((IMe(3)-M(3))^2));
% PULGAR
Dis(5) = sqrt(((PMe(1)-M(1))^2)+((PMe(2)-M(2))^2)+((PMe(3)-M(3))^2));
% ABIERTA
Dis(6) = sqrt(((ABMe(1)-M(1))^2)+((ABMe(2)-M(2))^2)+((ABMe(3)-M(3))^2));
% CERRADA
Dis(7) = sqrt(((CEMe(1)-M(1))^2)+((CEMe(2)-M(2))^2)+((CEMe(3)-M(3))^2));
[Vmin,Ind] = min(Dis);
% RESPUESTA ESTADISTICA
switch (Ind)
case 1
app.textoComparacion.Text = 'Estadistica: Menique';
case 2
app.textoComparacion.Text = 'Estadistica: Anular';
case 3
app.textoComparacion.Text = 'Estadistica: Corazon';
```

```
case 4
app.textoComparacion.Text = 'Estadistica: Indice';
case 5
app.textoComparacion.Text = 'Estadistica: Pulgar';
case 6
app.textoComparacion.Text = 'Estadistica: Abierta';
case 7
app.textoComparacion.Text = 'Estadistica: Cerrada';
end
drawnow
end
end


% Button pushed function: GuardarButton
function GuardarButtonPushed(app, event)
NomDedo = app.DedosButtonGroup.SelectedObject.Text;
if ~isempty(app.datosE1)
sdatos1 = app.datosE1;
sdatos2 = app.datosE2;
sdatos3 = app.datosE3;
app.nmuestra = app.nmuestra + 1;
app.MuestraSpinner.Value = app.nmuestra;
save(strcat('Dataset\',NomDedo,'\','SE1_',int2str(app.nmuestra),'.mat'),'sdatos1
');
save(strcat('Dataset\',NomDedo,'\','SE2_',int2str(app.nmuestra),'.mat'),'sdatos2
');
save(strcat('Dataset\',NomDedo,'\','SE3_',int2str(app.nmuestra),'.mat'),'sdatos3
');
app.datosE1 = [];
app.datosE2 = [];
app.datosE3 = [];
app.textoEstado.Text = 'Los datos han sido guardados satisfactoriamente.';
drawnow
else
app.textoEstado.Text = 'No hay datos para guardar. Grabe un movimiento e
inténtelo nuevamente.';
drawnow
end
end


% Button pushed function: EntrenarButton
function EntrenarButtonPushed(app, event)
app.textoEstado.Text = 'Entrenando...';
drawnow
% RECONOCIMIENTO
Entradas=[];
```

```
Salida=[];
for Ejemplos = 1:300
for nDedo = 1:7
switch nDedo
case 1
dedo = 'Menique';
Salida = [Salida [1;0;0;0;0;0;0]];
case 2
dedo = 'Anular';
Salida = [Salida [0;1;0;0;0;0;0]];
case 3
dedo = 'Corazon';
Salida = [Salida [0;0;1;0;0;0;0]];
case 4
dedo = 'Indice';
Salida = [Salida [0;0;0;1;0;0;0]];
case 5
dedo = 'Pulgar';
Salida = [Salida [0;0;0;0;1;0;0]];
case 6
dedo = 'Abierta';
Salida = [Salida [0;0;0;0;0;1;0]];
case 7
dedo = 'Cerrada';
Salida = [Salida [0;0;0;0;0;0;1]];
end
Aux=[];
for electrodo=1:3
selectrodo=int2str(electrodo);
datos=importdata(strcat('Electrodo_',selectrodo,'\',dedo,'.mat'));
Aux = [Aux,datos(Ejemplos,:)];
end
Entradas = [Entradas, Aux'];
end
end
app.MLP = ...
MultiLayerPerceptron('LengthsOfLayers', [21 7 7],...
'HiddenActFcn', 'tanh',...
'OutputActFcn', 'softmax',...
'UpdateFcn', 'default');
% Options
Options = ...
struct('TrainingAlgorithm', 'LM',...
'NumberOfEpochs', 60,...
'MinimumMSE', 1e-3,...
'InitialMarquardtParameter', 0.01,...
'MaximumMarquardtParameter', 1e10,...
'MarquardtFactor', 10);
```

```
% Training
app.MLP.train(Entradas,Salida,Options);
app.textoEstado.Text = 'Entrenamiento Finalizado.';
app.GrabarButton.Enable = 'on';
drawnow
end


% Callback function
function CerrarButtonPushed(app, event)
app.Entrenamiento.Visible = "off";
end


% Callback function
function FiltrarButtonPushed(app, event)
app.textoEntrenamiento.Text = 'Filtrando los datos...';
drawnow
end
end


% Component initialization
methods (Access = private)


% Create UIFigure and components
function createComponents(app)


% Create UIFigure and hide until all components are created
app.UIFigure = uifigure('Visible', 'off');
app.UIFigure.Position = [100 100 699 479];
app.UIFigure.Name = 'MATLAB App';


% Create GrabarButton
app.GrabarButton = uibutton(app.UIFigure, 'push');
app.GrabarButton.ButtonPushedFcn = createCallbackFcn(app, @GrabarButtonPushed,
true);
app.GrabarButton.Enable = 'off';
app.GrabarButton.Position = [213 58 106 22];
app.GrabarButton.Text = 'Grabar';


% Create Image
app.Image = uiimage(app.UIFigure);
app.Image.ScaleMethod = 'none';
```

```
app.Image.Position = [494 209 169 151];
app.Image.ImageSource = 'Abierta.png';


% Create Titulo
app.Titulo = uilabel(app.UIFigure);
app.Titulo.HorizontalAlignment = 'center';
app.Titulo.FontName = 'Impact';
app.Titulo.FontSize = 30;
app.Titulo.Position = [41 408 625 38];
app.Titulo.Text = 'Reconocimiento de Movimientos de la Mano';


% Create textoEstado
app.textoEstado = uilabel(app.UIFigure);
app.textoEstado.Position = [74 135 384 18];
app.textoEstado.Text = '¡Bienvenido!';


% Create DedosButtonGroup
app.DedosButtonGroup = uibuttongroup(app.UIFigure);
app.DedosButtonGroup.TitlePosition = 'centertop';
app.DedosButtonGroup.Title = 'Dedos';
app.DedosButtonGroup.Position = [516 15 123 177];


% Create MeniqueButton
app.MeniqueButton = uiradiobutton(app.DedosButtonGroup);
app.MeniqueButton.Text = 'Menique';
app.MeniqueButton.Position = [11 131 68 22];
app.MeniqueButton.Value = true;


% Create AnularButton
app.AnularButton = uiradiobutton(app.DedosButtonGroup);
app.AnularButton.Text = 'Anular';
app.AnularButton.Position = [11 109 65 22];


% Create CorazonButton
app.CorazonButton = uiradiobutton(app.DedosButtonGroup);
app.CorazonButton.Text = 'Corazon';
app.CorazonButton.Position = [11 87 67 22];


% Create IndiceButton
app.IndiceButton = uiradiobutton(app.DedosButtonGroup);
app.IndiceButton.Text = 'Indice';
```

```
app.IndiceButton.Position = [11 66 65 22];


% Create PulgarButton
app.PulgarButton = uiradiobutton(app.DedosButtonGroup);
app.PulgarButton.Text = 'Pulgar';
app.PulgarButton.Position = [11 45 65 22];


% Create AbiertaButton
app.AbiertaButton = uiradiobutton(app.DedosButtonGroup);
app.AbiertaButton.Text = 'Abierta';
app.AbiertaButton.Position = [11 24 60 22];


% Create CerradaButton
app.CerradaButton = uiradiobutton(app.DedosButtonGroup);
app.CerradaButton.Text = 'Cerrada';
app.CerradaButton.Position = [10 3 65 22];


% Create GuardarButton
app.GuardarButton = uibutton(app.UIFigure, 'push');
app.GuardarButton.ButtonPushedFcn = createCallbackFcn(app, @GuardarButtonPushed,
true);
app.GuardarButton.Position = [355 69 102 22];
app.GuardarButton.Text = 'Guardar';


% Create MuestraSpinnerLabel
app.MuestraSpinnerLabel = uilabel(app.UIFigure);
app.MuestraSpinnerLabel.HorizontalAlignment = 'right';
app.MuestraSpinnerLabel.Position = [354 39 49 22];
app.MuestraSpinnerLabel.Text = 'Muestra';


% Create MuestraSpinner
app.MuestraSpinner = uispinner(app.UIFigure);
app.MuestraSpinner.Position = [406 38 51 22];


% Create EntrenarButton
app.EntrenarButton = uibutton(app.UIFigure, 'push');
app.EntrenarButton.ButtonPushedFcn = createCallbackFcn(app,
@EntrenarButtonPushed, true);
app.EntrenarButton.Position = [73 58 106 22];
app.EntrenarButton.Text = 'Entrenar';
```

```
% Create textoComparacion
app.textoComparacion = uilabel(app.UIFigure);
app.textoComparacion.Position = [74 99 384 22];
app.textoComparacion.Text = '';


% Create Grafica
app.Grafica = uiaxes(app.UIFigure);
title(app.Grafica, 'Señal Registrada')
xlabel(app.Grafica, 'Dato de la muestra')
ylabel(app.Grafica, 'Potencia (W)')
app.Grafica.PlotBoxAspectRatio = [2.20529801324503 1 1];
app.Grafica.Position = [74 168 383 209];


% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end


% App creation and deletion
methods (Access = public)


% Construct app
function app = AppMyo


% Create UIFigure and components
createComponents(app)


% Register the app with App Designer
registerApp(app, app.UIFigure)


if nargout == 0
clear app
end
end


% Code that executes before app deletion
function delete(app)


% Delete UIFigure when app is deleted
```

```
delete(app.UIFigure)
end
end
end
```