ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

MODELADO DE LAS SEÑALES MIOELÉCTRICAS PRODUCIDAS POR LAS EXTREMIDADES SUPERIORES, USANDO MACHINE LEARNING, PARA PERSONAS CON NIVELES DE AMPUTACIÓN Y/O MALFORMACIÓN TRANSRADIAL.

Trabajo de Integración Curricular

Tipo: Proyecto Técnico

Presentado para optar al grado académico de:

INGENIERO/A EN ELECTRÓNICA Y AUTOMATIZACIÓN

AUTORES:

ALEXIS BAYARDO VACA BARAHONA

DAYANA ESTEFANÍA PAGUAY BADILLO

Riobamba – Ecuador

2024

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

MODELADO DE LAS SEÑALES MIOELÉCTRICAS PRODUCIDAS POR LAS EXTREMIDADES SUPERIORES, USANDO MACHINE LEARNING, PARA PERSONAS CON NIVELES DE AMPUTACIÓN Y/O MALFORMACIÓN TRANSRADIAL.

Trabajo de Integración Curricular

Tipo: Proyecto Técnico

Presentado para optar al grado académico de:

INGENIERO/A EN ELECTRÓNICA Y AUTOMATIZACIÓN

AUTORES: ALEXIS BAYARDO VACA BARAHONA

DAYANA ESTEFANÍA PAGUAY BADILLO

DIRECTOR: ING. RAMIRO FERNANDO ISA JARA. PhD

Riobamba – Ecuador

2024

Nosotros, Alexis Bayardo Vaca Barahona y Dayana Estefanía Paguay Badillo, declaramos que el presente Trabajo de Integración Curricular es de nuestra autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autores asumimos la responsabilidad legal y académica de los contenidos de este Trabajo de Integración Curricular; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 28 de marzo de 2024

Alexis Bayardo Vaca Barahona
0604385567

Dayana Estefanía Paguay Badillo
0604773283

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
CARRERA ELECTRÓNICA Y AUTOMATIZACIÓN

El Tribunal del Trabajo de Integración Curricular certifica que: El Trabajo de Integración Curricular; tipo: Proyecto Técnico, **MODELADO DE LAS SEÑALES MIOELÉCTRICAS PRODUCIDAS POR LAS EXTREMIDADES SUPERIORES, USANDO MACHINE LEARNING, PARA PERSONAS CON NIVELES DE AMPUTACIÓN Y/O MALFORMACIÓN TRANSRADIAL.** Realizado por los señores: **ALEXIS BAYARDO VACA BARAHONA Y DAYANA ESTEFANÍA PAGUAY BADILLO**, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Integración Curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	**FECHA**
Ing. Washington Gilberto Luna Encalada **PRESIDENTE DEL TRIBUNAL**	_____	2024-03-28
Ing. Ramiro Fernando Isa Jara. PhD **DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR**	_____	2024-03-28
Ing. Franklin Geovanni Moreno Montenegro **ASESOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR**	_____	2024-03-28

DEDICATORIA

Dedico este trabajo a mis seres más queridos: mi familia, amigos y aquellos que han sido fundamentales en mi camino académico. A mis amados padres, Bayardo Rogelio y Elizabeth Susana, les expreso mi más profundo agradecimiento por haberme brindado todo su apoyo incondicional, su sacrificio y sus invaluables enseñanzas que han sido la luz que ha guiado mi camino hacia la profesionalidad. En especial, a mi madre, quien con su amor infinito y su dedicación incansable ha sido mi inspiración y mi fuerza motriz. A mis hermanos, Bryan y Vicky, les reconozco como mi inspiración constante, mi apoyo y mi fortaleza. A mis abuelitos, Silvio, Rosa, Fanny y Rogelio, les agradezco por su cariño, sabiduría y consejos que han sido pilares en mi crecimiento. A toda mi familia, tíos y primos, les reconozco por su constante apoyo y amor, siempre presentes en mi vida, velando por mi bienestar y el de los suyos. A mis amigos, quienes han compartido conmigo risas, preocupaciones y momentos difíciles, les agradezco por estar siempre a mi lado, celebrando juntos cada logro y cada meta alcanzada. De manera especial, quiero dedicar este título a mi prima Kerly, quien ha sido mi compañera de aulas y mi apoyo incondicional. Cada logro y triunfo alcanzado lleva impreso el amor, la confianza y el aliento que todos ustedes me han brindado. Este trabajo va dedicado con profunda gratitud y cariño a todos ustedes, mis seres más amados.

Alexis

Quiero dedicar este trabajo a todas las personas que me apoyaron, creyeron y confiaron en mí, sin ustedes esto hubiese sido un poquito más difícil, pero gracias a su compañía todo fue mucho mejor y bonito.

Estefanía

AGRADECIMIENTO

Quiero expresar mi profundo agradecimiento a mi alma mater, la ESPOCH por brindarme una educación de calidad y un ambiente propicio para mi crecimiento académico. Agradezco a todos los docentes por su dedicación y compromiso con nuestra formación. En especial, quiero agradecer al Ing. Ramiro Isa, mi tutor, por sus orientaciones y guías indispensables que hicieron posible el desarrollo y culminación de este trabajo. Agradezco también a las autoridades de la facultad, al Decano Ing. Washington Luna, al Coordinador de la carrera Ing. José Morales y a la Secretaria Ximena Estrella, cuyo compromiso y dedicación han contribuido a crear un ambiente propicio donde nos han permitido expresarnos y desarrollarnos en ámbitos académicos y extraacadémicos. A mis padres, hermanos y familia en general, les agradezco profundamente por su apoyo incondicional, su amor y sacrificio, que han sido fundamentales en mi trayectoria académica. Este logro también es suyo. A todos ustedes, mi más sincero agradecimiento. Por último, quiero expresar mi agradecimiento a las autoridades institucionales, bajo el liderazgo del Ing. Byron Vaca, por su constante apoyo y gestión en beneficio de todos los estudiantes que, como yo, buscamos alcanzar nuestros sueños dentro de estas aulas. Su apoyo ha sido fundamental para la consecución de este logro.

Alexis

Primero quiero agradecer a Dios y la Virgencita por darme la oportunidad de finalizar una meta más, especialmente agradezco a mi papá José y mi mamá Mayra por estar en este camino y apoyarme en cada paso que estoy dando, ellos son mi motor junto con mis hermanos Pepe y Felipe. También quiero agradecer a mis amigos que estuvieron en esta etapa tan bonita de mi vida, por cada risa y grupo de trabajo que tuvimos, a mis abuelitos, tíos y primos que estuvieron en cada logro que alcance en esta bonita institución, a mi querido novio Joel por acompañarme y darme el apoyo para poder culminar con este objetivo, gracias a mi poli por darme muchas oportunidades para cumplir mis sueños, crecer y formarme como persona y profesional. A mi tutor Ing. Ramiro Isa por ayudarnos y no dejarnos solos en ningún momento, por confiar en nosotros. Y finalmente me quiero agradecer a mi porque con mucho esfuerzo, dedicación, confianza y amor, pude lograr uno de mis objetivos, demostrando lo fuerte y segura que soy en cada paso que doy.

Estefanía

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ANEXOS

RESUMEN

Actualmente, las personas que tienen algún tipo de amputación y/o malformación transradial se enfrentan a desafíos de movilidad mediante el uso de prótesis móviles como las mioeléctricas. Sin embargo, el control preciso de estas prótesis sigue siendo un desafío debido a la complejidad en la interpretación de las señales generadas en los músculos. Por tanto, el presente proyecto describe el modelado de señales mioeléctricas generadas por las extremidades superiores durante la ejecución de 8 actividades predefinidas, con el objetivo de mejorar los resultados de la clasificación para el control de prótesis para personas con esta discapacidad. Para ello, se usa una tarjeta de desarrollo ESP32S y tres sensores MyoWare integrados en una manilla que se ubican en tres músculos diferentes: tensor, bíceps superior e inferior. La señal mioeléctrica adquirida proporciona valores de 0V a 5V, los cuales se almacenan en archivos CSV de acuerdo con cada canal. Esto permite obtener una base de datos para entrenar y validar el modelo de Machine Learning usando Google Colab. Los datos son normalizados y estandarizados para el entrenamiento de una red neuronal convolucional (CNN) para clasificación de 8 categorías. La división de los datos fue del 80% para entrenamiento y del 20% para prueba. La red categórica alcanzó el 100% de precisión en cinco categorías y de 98,6%; 98,8% y 98,5% en las tres restantes. El modelo de predicción temporal logra una correlación cercana al 98% en el 95% de las señales con respecto a la señal original. Las CNNs identifican patrones complejos que permiten clasificar los diferentes tipos de movimientos y predecir su comportamiento futuro con una alta precisión. Finalmente, el proyecto destaca el potencial del Machine Learning y de las redes neuronales en el control de prótesis funcionales, contribuyendo al avance de la rehabilitación física para personas con esta discapacidad.

Palabras clave: <SEÑALES MIOELÉCTRICAS>, <MYOWARE (HARDWARE)>, <PYTHON (SOFTWARE)>, <REDES NEURONALES CONVOLUCIONALES (CNN)>, <MACHINE LEARNING>, <CLASIFICACIÓN>.

SUMMARY

Currently, people who have some transradial amputation and/or malformation face mobility challenges through the use of mobile prostheses such as myoelectric ones. However, the precise control of these prostheses remains challenging due to the complexity of interpreting the signals generated in the muscles. Therefore, the present project describes the modeling of myolectric signals generated by the upper limbs during the execution of eight predefined activities to improve the results of the classification for prosthetic control for people with this disability: To do this, an ESP32S development board and three MyoWare sensors integrated into a wristband are used, which are placed on three different muscles: tensor, upper and lower biceps. The acquired myoelectric signal provides values from 0V to 5V, which are stored in CSV files for each channel. This allows for obtaining a database to train and validate the machine learning model using Google Colab. The data is normalized and standardized to train a convolutional neural network (CNN) to classify eight categories. The data was split into 80% for training and 20% for testing. The categorical network reached 100% accuracy in five categories: 98.6%, 98.8%, and 98.5% in the three remaining categories. The temporal prediction model achieves a correlation close to 98% in 95% of the signals concerning the original signal. CNNs identify complex patterns that classify different types of movements and predict their future behavior with high accuracy. Finally, the project highlights the potential of Machine Learning and neural networks in controlling functional prostheses, contributing to the advancement of physical rehabilitation for people with this disability.

Keywords: <MYOELECTRICAL SIGNALS>, <MYOWARE (HARDWARE)>, <PYTHON (SOFTWARE)>, <CONVOLUCIONAL NEURONAL NETWORKS (CNN)>, <MACHINE LEARNING>, <CLASIFICATION>.

_____

Lenin Iván Lara Olivo
0602546103

INTRODUCCIÓN

La discapacidad física motora afecta a millones de individuos en todo el mundo. La pérdida de las extremidades superiores debido a amputación y/o malformación representa un desafío significativo en términos de autonomía, funcionalidad y calidad de vida. Para ello, se han desarrollado diversas soluciones con el fin de enfrentar este desafío una de ella son las prótesis (Suárez García, 2021). Existen de dos tipos: las pasivas o no funcionales que tienen características similares a las extremidades no faltantes como el tono de piel, tamaño y forma; y por otro lado las prótesis móviles presentan varias ventajas con relación a sus movimientos y autonomía; Dentro de este grupo están las prótesis mioeléctricas que se activan a través de señales electromiográficas o musculares, las cuales han surgido como una solución innovadora para ayudar la funcionalidad de las extremidades afectadas. Además, representan un avance significativo en la rehabilitación y el empoderamiento de las personas (Linares & Rosas, 2019).

El uso de tecnologías avanzadas como la inteligencia artificial específicamente el Machine Learning (ML) para analizar, interpretar y predecir los movimientos a partir de señales EMG se presenta como una solución prometedora para mejorar la precisión y eficiencia de las prótesis funcionales. Este enfoque no solo busca optimizar el funcionamiento y control de las prótesis, sino también mejorar la calidad de vida de los usuarios, al brindarles la capacidad de interacción con su entorno, independencia en sus actividades diarias y algunos casos mejorar su autoestima.

De esta manera, por medio de la adquisición, procesamiento y almacenamiento de los datos provenientes de las señales mioeléctricas, se realiza un entrenamiento a un modelo de ML basado en redes neuronales convolucionales, se pueden modelarlas para posteriormente predecir los movimientos que puede realizar una prótesis funcional. Esto tiene relevancia dentro del campo tecnológico y de ingeniería aplicado a la salud en las áreas de rehabilitación para el control de prótesis para extremidades superiores y potenciar así las oportunidades a los usuarios con amputación y/o malformación transradial.

CAPÍTULO I

1. PLANTEAMIENTO DEL PROBLEMA

1.1 Planteamiento del problema

1.1.1 Antecedentes

Según (Suárez García, 2021), menciona que alrededor del mundo existen más de mil millones de personas que sufren de alguna discapacidad. Dentro de dichas personas se encuentran un segmento de personas que han sufrido amputación o malformación de alguna o ambas extremidades superiores. Para contrarrestar este problema existen tres tipos de prótesis que son prótesis estéticas que solo interesa la apariencia física; las prótesis funcionales pasivas las cuales se ajustan en movimientos de prensión específicos y por último, las prótesis funcionales activas donde ellas utilizan energía externa y pueden ser mecatrónicas y además están controladas por medio de señales bioeléctricas. Una subsección de estas prótesis funcionales son las prótesis mioeléctricas, dicha prótesis según (Gigli et al., 2023) pueden devolver o aumentar la independencia de las personas con diferencias en las extremidades, permitiéndoles realizar diversas actividades de la vida diaria. Estas prótesis pueden sustituir a una extremidad superior humana tras una amputación o una deficiencia con el objetivo de restaurar el aspecto estético y la función. (Kerver et al., 2023). La electromiografía (EMG) es el proceso de detectar las señales eléctricas generadas por los músculos durante una contracción muscular. Esta técnica implica registrar y analizar la actividad eléctrica que se produce en los nervios y los músculos mediante el uso de electrodos, ya sea superficiales, de aguja o implantados. Las mediciones obtenidas a través de la EMG brindan información importante sobre la fisiología y los patrones de activación de los músculos (Crawford & Vavra, 2016). Sin embargo, debido a la complejidad que supone la interpretación de dichas señales y posterior a ello, el grado de exactitud para controlar el movimiento de las prótesis funcionales es bajo. Algunos investigadores han indicado que mediante un entrenador virtual se podría evaluar la adecuación de una prótesis mioeléctrica para personas con niveles de amputación y/o malformaciones transradial y posibilitar un aprendizaje previo de manejo de prótesis funcionales.

1.1.2 Formulación del problema

¿Se puede realizar un modelamiento de señales mioeléctricas producidas por la extremidad superior usando Machine Learning para personas con niveles de amputación y/o malformación transradial?

1.1.3 Sistematización del problema

- ¿Por qué realizar un estudio de señales mioeléctricas, incluyendo la investigación y selección de los componentes de hardware y software necesarios para la adquisición, procesamiento y análisis de dichas señales?
- ¿Para qué elaborar un sistema de adquisición de señales mioeléctricas mediante la integración de tres sensores para obtener los datos del sistema de inteligencia computacional mediante un entorno de software adecuado?
- ¿Se puede desarrollar un sistema de entrenamiento de Machine Learning para la interpretación y predicción de movimientos en prótesis funcionales?
- ¿Cómo realizar pruebas y validaciones del sistema?

1.2 Justificación

1.2.1 Justificación teórica

La discapacidad física motora es un problema mundial, su afectación incluye la reducción del control y movimiento del cuerpo de la persona, llegando a restringir el desarrollo personal y social, al verse afectado su libre desplazamiento, la capacidad de manipular objetos, dificultades para sostener posturas y para mantener el equilibrio corporal, la comunicación y la respiración (Bravo Loor, 2023).

La rehabilitación física desempeña un papel fundamental en la mejora de la calidad de vida de las personas con discapacidad física. En los últimos años, se ha observado un notable incremento en el uso de tecnologías de las personas que han perdido una extremidad o han nacido sin estas, pero todas apuntan a un solo futuro, la implementación de aditamentos electromecánicos para su operación, desde servomotores, baterías e impresión en 3D. (Linares & Rosas, 2019). Estas tecnologías continúan mejorando constantemente, principalmente en la implementación de técnicas de control avanzado, donde nos permite un procesamiento mucho más inteligente y también eficiente. Obteniendo como resultado el aumento significativamente de la precisión al tener movimientos en el sistema y a su vez, por parte de los usuarios una mejor respuesta a la hora de adquirirlo. Esto implica desarrollar un entrenador mioeléctrico virtual para evaluar la adecuación de una prótesis de señales mioeléctricas para amputados de mano y posibilitar un aprendizaje previo de manejo de prótesis.(Garnica, 2020).

Comúnmente para realizar este tipo de investigaciones son usadas las señales electroencefalografía (EEG), ya que se resaltan por generar sistemas con mayor precisión la intención motora del usuario, sin embargo existen inconvenientes inherentes por lo que la actividad cerebral genera patrones de voltaje muy pequeños, además estas señales son susceptibles al ruido electromagnético que las señales EMG, por estos motivos las señales electromiográficas son las más recomendades para el desarrollo de un sistema de

entrenamiento.(Sánchez Granja, 2021). Pero, estas señales también conllevan desafíos adicionales que requieren atención. Para lograr un modelamiento preciso y confiable, es necesario contar con una amplia base de datos que contenga muestras de señales mioeléctricas de un gran grupo de personas las cuales nos ayuden a que el sistema sea más consistente y certero.

Además, se debe asegurar la robustez del sistema de entrenamiento, lo que implica desarrollar algoritmos y técnicas capaces de manejar diferentes condiciones y escenarios, ya sea con el sensor mioeléctrico como pueden ser variaciones con la colocación de los electrodos o con interferencias electromagnéticas (Vázquez et al., 2020). Teniendo en cuenta varios aspectos, se garantizará que el sistema funcione de manera segura y precisa al poner en práctica, donde el sistema sea capaz de que los pacientes se enfrenten a diversas situaciones y desafíos.

Otro aspecto crítico es la adaptación del sistema a las necesidades y habilidades cambiantes de los usuarios a lo largo del tiempo. Las capacidades motoras de los usuarios pueden evolucionar con el tiempo, por lo tanto, el sistema debe ser capaz de adaptarse y ajustarse a estas necesidades en un cambio constante (Madou, 2020).

Las ventajas de utilizar señales mioeléctricas para el modelamiento de un sistema de entrenamiento son múltiples. Conceden una interacción directa entre el sistema y las extremidades superiores del paciente o usuario, lo que conlleva a una mayor firmeza del sistema para así, poder adaptarse y predecir posibles ordenes de control y ejecutarlas con mayor eficiencia (Schlotthauer , 2022). Por otro lado, este modelamiento con señales mioeléctricas de las extremidades superiores ofrece una conveniencia para mejorar las sensaciones y experiencias percibidas por las personas con discapacidad física motora.

1.2.2 Justificación aplicativa

La ausencia de extremidades en este caso de un antebrazo repercute a la capacidad de las personas afectadas para realizar actividades de la vida diaria, lo que conlleva un impacto significativo en su independencia y calidad de vida. (Castro, 2022). Las amputaciones o malformaciones no solo conllevan desafíos físicos si no también emocionales y psicológicos (Madou, 2020).

Es por ello por lo que se decide usar la tecnología como una oportunidad para mejorar la calidad de vida de estas personas y restaurar parcialmente la funcionalidad perdida mediante técnicas innovadoras y cada vez más precisas como bien puede ser el modelamiento de señales y entrenamiento virtual del mismo.

El trabajo con señales mioeléctricas son una fuente importante de información neuronal que puede utilizarse para controlar las prótesis de extremidades superiores, toman estas señales de la parte residual del cuerpo o muñón para accionarlas. (Esaa et, 2023). Sin embargo, dichas técnicas de

control aún siguen siendo un desafío a causa de la inherente variabilidad de señales y demás complejidades al momento de intentar activar estos movimientos.

Este proyecto de tesis formará parte del Proyecto de Investigación Multidisciplinario IDIPM001, el cual se centra en la creación de prótesis personalizadas divididas en tres fases: prótesis estéticas, prótesis con activación mioeléctrica y prótesis con activación de señales cerebrales. Este proyecto busca abordar las necesidades de las personas con amputación o malformación de brazo, brindando soluciones tecnológicas innovadoras para mejorar su calidad de vida y autonomía.

El propósito fundamental de este proyecto es desarrollar un modelo de Machine Learning capaz de interpretar las señales mioeléctricas generadas por las extremidades superiores de personas con amputación o malformación de brazo. Estas señales son vitales para el control de prótesis y dispositivos de asistencia, ya que permiten traducir las intenciones del usuario en movimientos precisos de la prótesis. La aplicación exitosa de este modelo se traducirá en un control más intuitivo y preciso de las prótesis, lo que mejorará significativamente la calidad de vida de las personas afectadas.

Además, este proyecto contribuirá al avance de la investigación en la interacción entre señales biológicas y tecnologías de asistencia. La capacidad de interpretar y utilizar señales mioeléctricas abre nuevas puertas a la rehabilitación y la integración social de las personas con amputación de brazo. Esto significa que no solo se trata de proporcionar una herramienta funcional, sino de empoderar a los individuos para que recuperen su independencia y participen activamente en la sociedad.

1.3 Objetivos

1.3.1 Objetivo General

Modelar las señales mioeléctricas producidas por las extremidades superiores usando Machine Learning para personas con amputación y/o malformación transradial.

1.3.2 Objetivos Específicos

- Estudiar las señales mioeléctricas como se generan, sus características eléctricas.
- Determinar los componentes de hardware y software necesarios, para la adquisición, procesamiento y análisis de señales mioeléctricas.
- Diseñar e implementar un sistema de adquisición de señales mioeléctricas,
- Desarrollar un sistema de entrenamiento de Machine Learning para la interpretación y predicción de movimientos en prótesis funcionales.
- Validar el funcionamiento del sistema implementado, a través de las pruebas respectivas.

CAPÍTULO II

2. MARCO TEÓRICO

2.1 Señales

Una señal es una representación cuantitativa o cualitativa de un fenómeno físico, acústico, visual, eléctrico, etc. Estas pueden adoptar diversas formas, incluyendo ondas eléctricas, sonidos, impulsos lumínicos e imágenes entre las más comunes. Estas señales son utilizadas en numerosas aplicaciones dentro de la ingeniería, la ciencia, las telecomunicaciones entre otras. (Aguagallo, 2019).

El procesamiento de señales involucra la adquisición, análisis, y representación de la información, lo cual, es fundamental para el control de sistemas, detección de patrones, transmisión de datos y otras aplicaciones dentro del campo tecnológico.

Las señales eléctricas, piedra angular dentro de la ingeniería electrónica, ingeniería eléctrica y de las comunicaciones, se definen por su dinámica de corriente o tensión eléctrica en función del tiempo. El procesamiento de estas señales involucra la aplicación de métodos y algoritmos destinados a optimizar, analizar y depurar la información contenida en ellas. Este involucra la amplificación, filtrado y conversión entre representaciones analógicas y digitales de señales, hasta la implementación de complejos algoritmos para el procesamiento de señales digitales (DSP). (Hernández Montejano, 2022). En la Ilustración 2-1 se puede observar diferentes formas de ondas de las señales eléctricas.

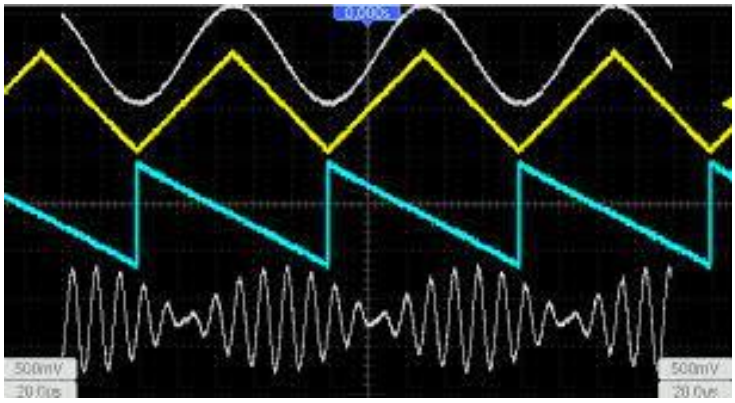


Ilustración 2-1 Señales eléctricas

Fuente: https://microchipotle.com/tipos-de-senales-electricas-y-electronicas/

2.1.1 Tipos de señales

2.1.1.1 Señales Analógicas (Tiempo Continuo)

Es la representación de datos que varían de manera continua en el tiempo. Pueden tomar muchas formas, como ondas senoidales, cuadradas o triangulares. En un sistema analógico, la información

se codifica mediante la amplitud y/o la frecuencia de la señal. (Navarro, 2021). En la Ilustración 2-2 se representa la señal analógica.

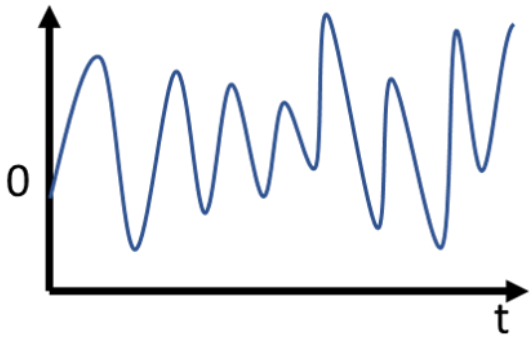


Ilustración 2-2 Señal analógica

Fuente: https://microchipotle.com/tipos-de-senales-electricas-y-electronicas/

2.1.1.2 Señales Digitales (Tiempo Discreto)

Se representan datos que toman valores discretos o distintos niveles en lugar de valores continuos en el tiempo, en estas señales la información se codifica utilizando un conjunto finito de estados. Estos estados se usan para transmitir, almacenar y procesar información en sistemas digitales. (Navarro, 2021). La manera habitual de obtener una representación discreta en el tiempo de una señal continua es tomando muestras cada determinando período de tiempo. Este proceso se puede realizar mediante series de Fourier, entre otros métodos. El muestreo, produce un espectro muy parecido a la señal continua, pero se despliega en una colección de réplicas del espectro fundamental, desplazada a intervalos que son múltiplos de la frecuencia de muestreo como se observa en la Ilustración 2-3. (Barco, 2013).

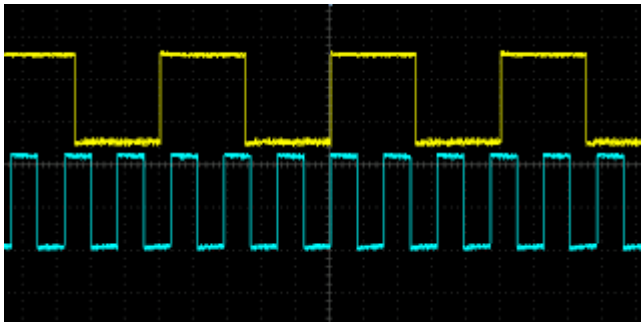


Ilustración 2-3 Señales digitales

Fuente: https://microchipotle.com/tipos-de-senales-electricas-y-electronicas/

En la Tabla 2-1, se presentan varios tipos de señales relevantes en el análisis de señales, con una breve descripción de cada una. Este panorama proporciona una visión general de la diversidad de señales en la práctica y su importancia en campos como la comunicación, el control de sistemas y la biomedicina.

Tabla 2-1 Clasificación y descripción de señales comunes

Tipo de Señal	Descripción
Señales Periódicas	Se repiten regularmente en un intervalo de tiempo, con un periodo constante, lo que supone que su forma se replicará en intervalos regulares a través del tiempo (IvanAg, 2019)..
Señales Determinísticas	Su comportamiento puede predecirse completamente utilizando funciones o expresiones matemáticas específicas. Su forma y valores futuros se conocen con certeza.
Señales Moduladas	Surgen a partir de una alteración en alguna de sus características para transportar información de manera eficiente según la aplicación en que se la emplea (Schlotthauer 2021).
Señales de Control	Utilizadas en sistemas de control para influir en el comportamiento de un sistema en general. Son generadas por controladores o sistemas de control automático.
Señales Bioeléctricas	Originadas por la actividad eléctrica en células, tejidos u órganos de organismos vivos. Pueden ser registradas mediante equipos especializados como EEG, ECG, EMG (Madou, 2020).

Realizado por: Vaca-Paguay, 2024.

2.2 Músculos del ser humano

Los músculos son un tejido que tiene como una de sus características básicas la capacidad de contraerse (Fernández, 2022). Histológicamente, los músculos se dividen en liso y estriado de acuerdo con la organización de su citoesqueleto. Las principales funciones de los músculos están vinculadas a la generación de movimiento, ya sea de las vísceras no cardiacas a través del músculo liso, el corazón a partir del músculo cardiaco o del sistema osteoarticular a partir de la contracción del músculo esquelético. Por este mismo mecanismo, estabilizan las posiciones corporales y participan directamente en el equilibrio y la propiocepción. (Luque, 2021). En la Ilustración 2-4 se puede ver todos los músculos del cuerpo humano.

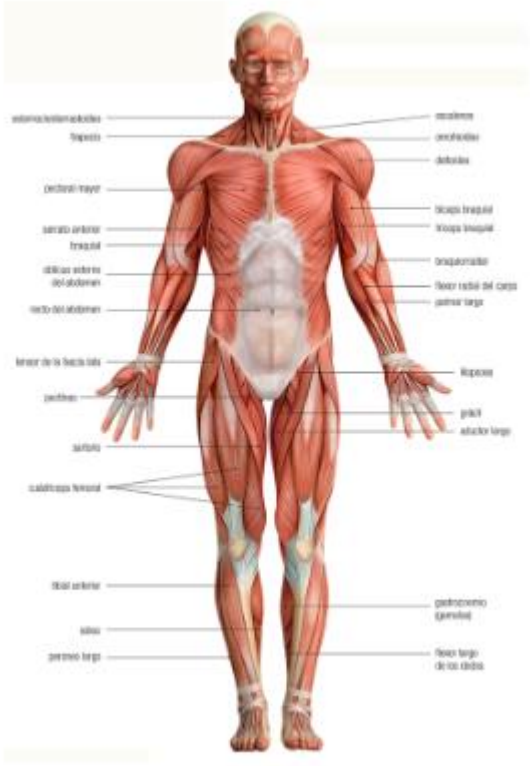


Ilustración 2-4 Músculos del cuerpo humano

Fuente: https://elibro.net/es/ereader/espoch/123771?page=12

2.2.1 Músculos de las extremidades superiores

2.2.1.1 Músculos bíceps

El bíceps braquial es un músculo que tiene dos cabezas o dos porciones a) la interna que es la cabeza larga y b) la externa que es la cabeza corta (Ferreira-Arquez, 2020). Su inserción permite la flexión o supinación del codo como se muestra en la Ilustración 2 -5. Con el antebrazo parcialmente flexionado, el bíceps es un potente supinador del antebrazo. Además, ayuda a otros músculos, aunque débilmente, como flexor del brazo a nivel del hombro. Se puede palpar este musculo bajo el deltoides que se encuentra aplastado contra la cabeza humeral. (Ferreira-Arquez, 2020)

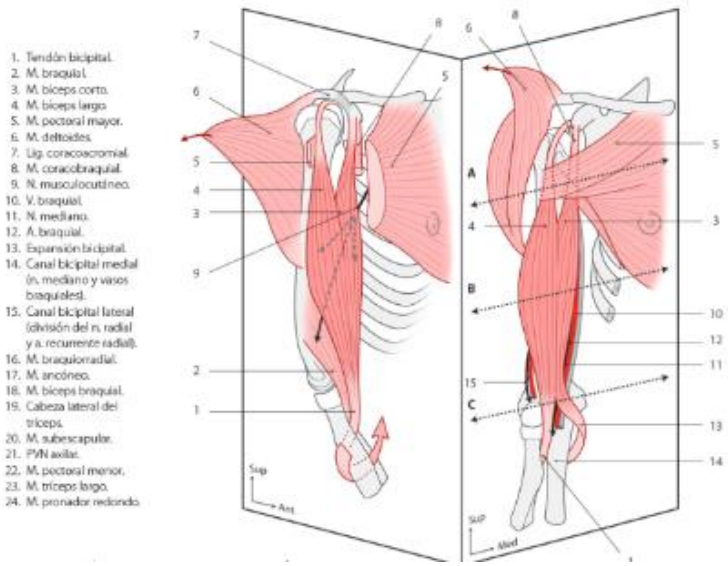


Ilustración 2-5 Músculo bíceps a) cabeza corta N°3 b) cabeza larga N°4

Fuente: (Dufour & Valle, 2020).

2.2.1.2 Músculo tríceps

El músculo tríceps braquial está formado por el músculo largo y cabeza medial y lateral. Su función principal es extender la articulación del codo, donde la cabeza larga también tiene la función de aducción y extensión de la articulación del hombro. (He et al., 2022). En la Ilustración 2-6 se observa el músculo del tríceps.

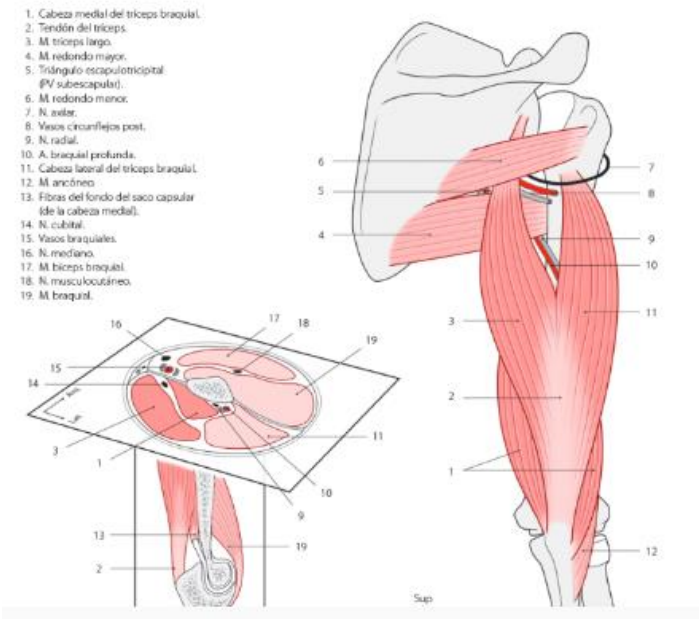


Ilustración 2-6 Musculo tríceps (N° 3)

Fuente: (Dufour & Valle, 2020).

2.2.1.3 Músculo braquial

El músculo braquial es uno de los principales flexores del antebrazo en la articulación del codo. Tiene una forma fusiforme (de huso) y se localiza en el compartimento braquial anterior (flexor), por debajo del músculo bíceps braquial. El músculo braquial se caracteriza por ser un músculo ancho, con su porción más grande ubicada en su punto medio en lugar de sus extremos. Además, su zona palpable tiene en el tendón ancho a ambos lados del bíceps y en la parte medial y, sobre todo, lateral del cuerpo carnoso (Dufour & del Valle , 2020). En la Ilustración 2-7 está representado el músculo braquial.

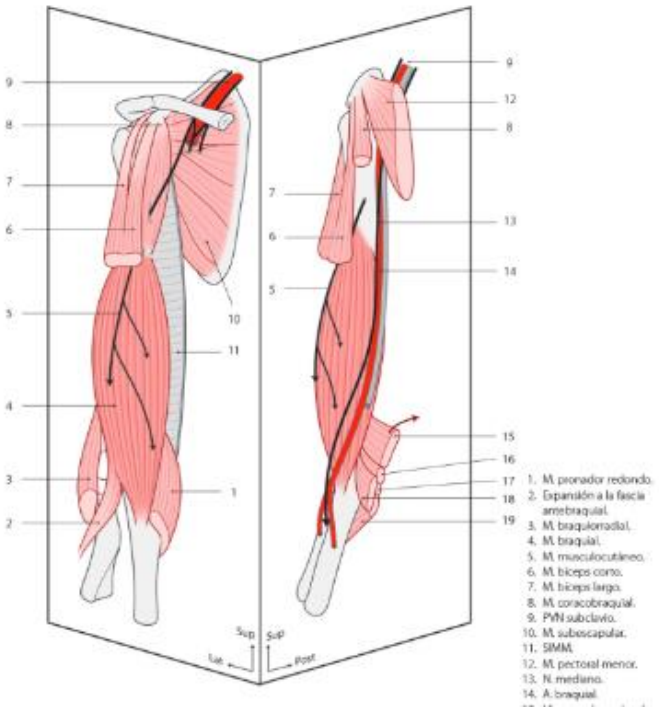


Ilustración 2-7 Musculo braquial (N° 4)

Fuente: (Dufour & Valle, 2020).

2.2.1.4 Músculo coracobraquial

El músculo coracobraquial es bastante fino y bordea por dentro del bíceps corto mediante un tendón común. Se le puede palpar desde el coracoides hasta la terminación del humeral, bordeando el bíceps corto, del que se diferencia por la ausencia de contracción en supinación. Este músculo facilita los movimientos del brazo y del hombro, ya que permite que el húmero pueda dirigirse hacia adelante y en dirección medial. También hace posible que los huesos del miembro superior puedan volver a su posición fisiológica (Dufour & del Valle , 2020). En la Ilustración 2-8 se encuentra el músculo coracobraquial.

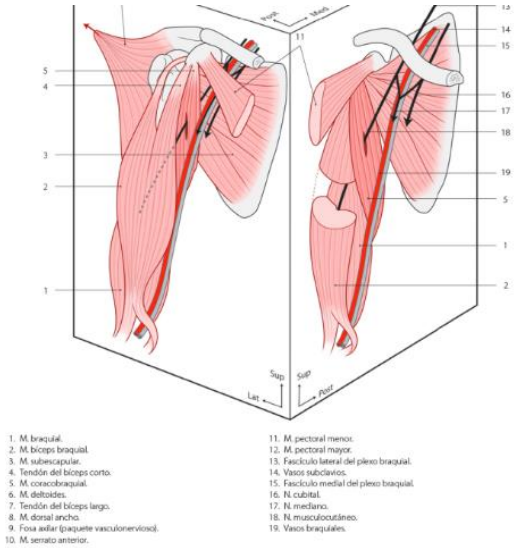


Ilustración 2-8 Músculo coracobraquial (N° 5)

Fuente: (Dufour& Valle, 2020).

2.3 Señales medibles a nivel muscular

Las señales musculares, o señales generadas por la actividad muscular, desempeñan un papel crucial en campos como la medicina, la rehabilitación y la ingeniería biomédica. Estas señales proporcionan información valiosa sobre la función muscular y se utilizan en el diagnóstico de trastornos neuromusculares, la evaluación del rendimiento deportivo, la rehabilitación y el desarrollo de tecnologías avanzadas para el control de prótesis y dispositivos tecnológicos (Azpilcueta, 2020).

2.3.1 Señales de presión muscular

Se pueden cuantificar mediante sensores de presión que se aplican sobre la superficie de la piel o que se insertan directamente en el músculo. Estos sensores registran las variaciones de presión generadas durante la contracción muscular como se puede visualizar en la Ilustración 2-9.

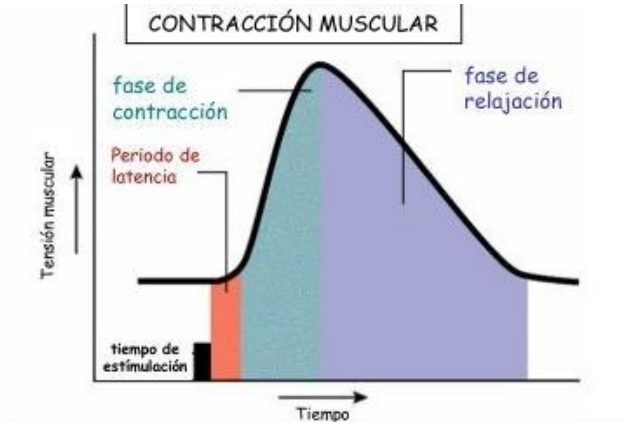


Ilustración 2-9 Señal de una contracción muscular

Fuente: https://www.yumpu.com/es/document/read/14421058/contraccion-muscular-todoenfermeria

2.3.2 Señales de temperatura muscular

La temperatura de un músculo puede fluctuar en respuesta a la actividad muscular y a los cambios en el flujo sanguíneo. Para registrar estas variaciones, se emplean sensores de temperatura adecuados (Dalcame, 2024). Esta información resulta especialmente pertinente en aplicaciones vinculadas a la fisioterapia y al seguimiento de la respuesta del organismo al ejercicio como se observa en la Ilustración 2-10.

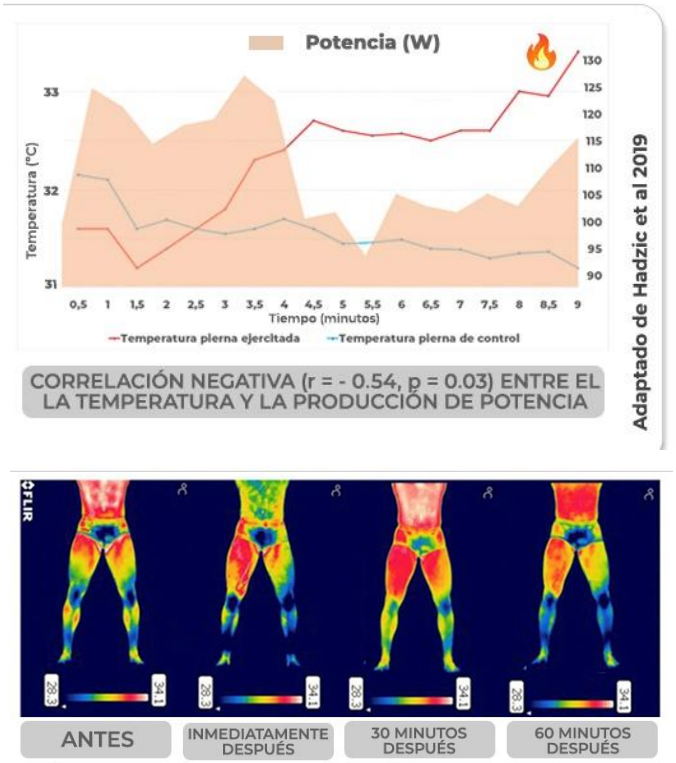


Ilustración 2-10 Ejemplo de señal de temperatura muscular con una pierna recién ejercitada

Fuente: (Estal & Escamilla, 2022)

2.3.3 Señales de oxigenación muscular

La oxigenación muscular hace referencia al aporte de oxígeno a los músculos. Se mide mediante oxímetros de pulso colocados en proximidad al músculo de interés (Vasquez, 2021). Estas mediciones resultan cruciales en contextos como el seguimiento de la oxigenación durante la realización de ejercicios como se puede ver en la Ilustración 2-11 y en procesos de rehabilitación.

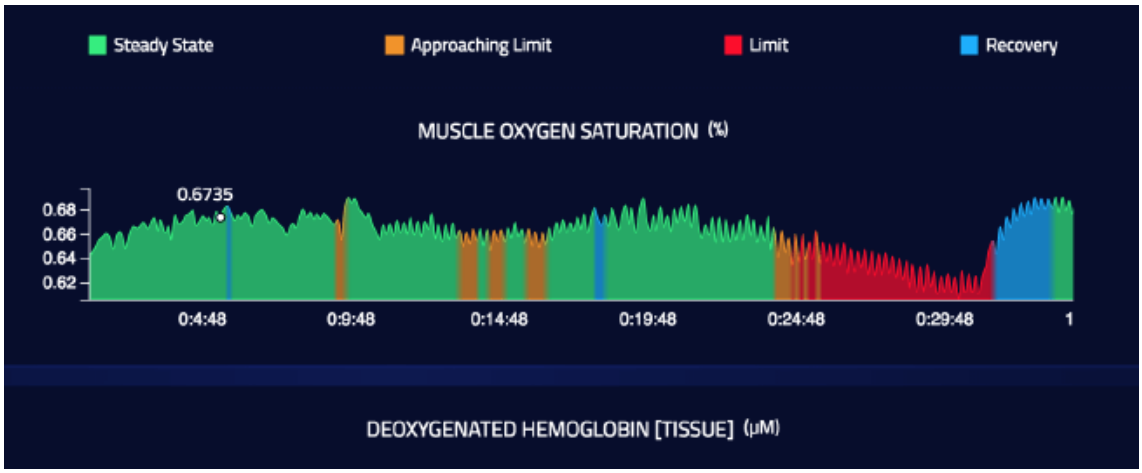


Ilustración 2-11 Señal de oxigenación muscular mientras realiza ejercicio

Fuente: https://carloslopezmartinez.com/humon-hex-saturacion-oxigeno-muscular/

2.3.4 Señales Mioeléctricas

Comúnmente conocidas como señales mioeléctricas superficiales (SEMG), se originan durante la actividad eléctrica de las fibras musculares presentes en la superficie de la piel. Estas señales se producen como resultado de la compleja interacción entre las neuronas motoras y las fibras musculares en momentos de contracción muscular (Schlotthauer, 2022).

Cuando una persona ejecuta un movimiento, las neuronas motoras transmiten impulsos eléctricos a las fibras musculares, lo que desencadena su despolarización y, como consecuencia, la generación de potenciales de acción. Estos potenciales dan lugar a una corriente eléctrica que se propaga a través del tejido muscular y finalmente emerge en la superficie de la piel, donde se la puede capturar mediante el uso de electrodos superficiales aplicados sobre la epidermis.(Azpilcueta, 2020)

En esencia, las señales mioeléctricas se presentan como representaciones eléctricas de la actividad muscular, ofreciendo valiosa información acerca de la activación, la intensidad y la sincronización de las fibras musculares. (Del Brío, 2020). En la ilustración 2-12 se puede observar el espectro Fourier de sEMG para 4 canales.

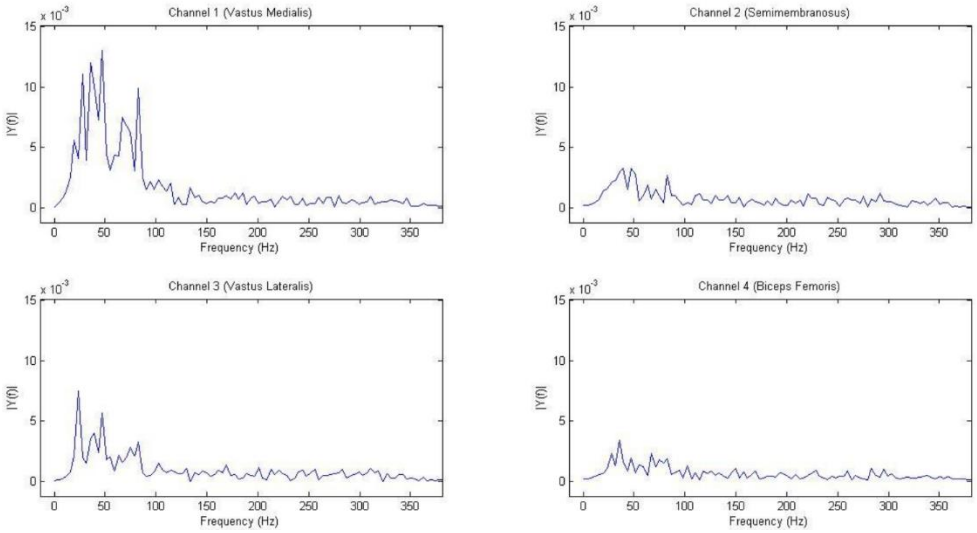


Ilustración 2-12 Espectro de Fourier para los 4 canales de sEMG

Fuente:(Cifuentes, 2012)

En el campo de la ingeniería biomédica y la fisiología muscular, es crucial comprender las señales utilizadas para analizar la actividad muscular. La Tabla 2-2 presenta tipos de señales musculares y sus descripciones, ofreciendo una visión general de su importancia en la rehabilitación y el rendimiento deportivo.

Tabla 2-2 Tipo de señales musculares

Tipo de Señal	Descripción
Señales Electromiográficas (EMG)	Se producen por la actividad de las células musculares durante la contracción o relajación muscular. La amplitud y forma de onda indican la intensidad y frecuencia de la actividad muscular (Vázquez, 2020).
Señales de Potenciales de Acción Muscular (MAP)	Generadas en las fibras musculares durante la contracción muscular. Reflejan la actividad eléctrica en las células musculares durante la excitación del músculo.
Señales de Fuerza Muscular (DIM)	Representan la magnitud de la fuerza ejercida por un músculo o grupo de músculos durante la contracción. Se utilizan para medir la fuerza en el rendimiento deportivo y la rehabilitación.
Señales Electromiográficas (EMG)	Se producen por la actividad de las células musculares durante la contracción o relajación muscular. La amplitud y forma de onda indican la intensidad y frecuencia de la actividad muscular.

Realizado por: Vaca-Paguay, 2024.

2.3.5 Frecuencia de Muestreo para Señales EMG

Para registrar adecuadamente la actividad muscular en señales EMG mediante equipos electrónicos, es importante que la frecuencia de muestreo sea suficientemente alta para capturar los detalles relevantes. Se sugiere utilizar una frecuencia de al menos 1000 muestras por segundo (1 kHz) para obtener datos precisos que se pueden utilizar en aplicaciones médicas o de investigación (Azpilcueta, 2020)

Una frecuencia de muestreo más alta proporciona una mejor resolución temporal, lo que significa que se pueden capturar detalles más finos en la señal, como cambios rápidos en la actividad muscular. Sin embargo, una frecuencia excesivamente alta puede resultar en un volumen de datos extenso que puede requerir mayor capacidad de procesamiento y almacenamiento. (Vasconcellos, 2022). En la Tabla 2-3 se puede ver la aplicación de diferentes rangos de frecuencia.

Tabla 2-3 Rangos de Frecuencia de Muestreo para Señales EMG.

Frecuencia de Muestreo	Aplicativo
200 Hz - 2000 Hz	Rango común: Permite capturar la actividad muscular en tiempo real con suficiente detalle.
1000 Hz - 4000 Hz	Mayor precisión: Adecuado para aplicaciones de investigación o diagnóstico que requieren alta resolución.
Hasta 10 kHz o más	Estudios avanzados: Para análisis detallados y experimentos especializados con altos requisitos de precisión.

Realizado por: Vaca-Paguay, 2024.

2.4 Niveles de amputación y/o malformación

La amputación se define como la supresión quirúrgica o la pérdida de una parte del cuerpo. Por ejemplo, la supresión total o parcial de las extremidades superiores o inferiores. Existen diferentes niveles de amputación en los miembros superiores como son los dedos, parte de la mano, articulación de la muñeca, transradial, articulación del codo, transhumeral, articulación del hombro y escapulotorácica (Michaud, 2022)

Las amputaciones y las malformaciones son irreversibles. Aunque existen formas de sustituir al miembro perdido, pero hasta el momento no funciona como un miembro biológico.

Después de una amputación o malformación se tiene una parte residual de la extremidad, llamada muñón. Se debe tener en cuenta el nivel de amputación o malformación para obtener una mayor funcionalidad del muñón, ya que puede contar con articulaciones móviles y con una musculatura

capaz de realizar movimientos como brazo de palanca adecuado para controlar una prótesis. (Vera Remache, 2021)

Según la Tabla 2-4, se puede observar la topográfica de Schwartz donde existen diversos niveles de amputación que son:

Tabla 2-4 Niveles de amputación o malformación en miembros superiores**.**

MIEMBROS SUPERIORES
Interescapulotoráxico
Desarticulación de Hombro
Amputación por arriba de codo
Desarticulación de codo
Amputación por debajo del codo
Desarticulación de muñeca
Amputaciones parciales de mano

Realizado por: Vaca-Paguay, 2024.

Se puede observar en la Ilustración 2-13, los diferentes tipos que pueden existir para la amputación o malformación y su ubicación en los miembros superiores del cuerpo del ser humano.

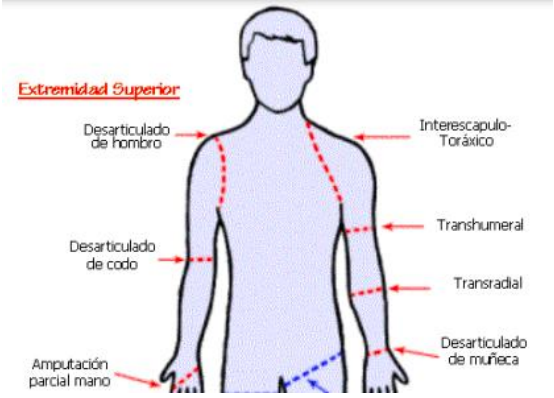


Ilustración 2-13 Nivel de amputaciones y/o malformaciones

Fuente: (Linares & Rosas, 2019)

2.4.1 Amputación o malformación transradial

La amputación o malformación transradial se ubica por debajo del codo y se trata de un corte a través de los huesos radio y cúbito. Dependiendo del nivel al que se encuentre la amputación o la malformación, se tiene la funcionalidad del muñón, ya que a niveles determinados se tendrá mayor o menor movimiento. (Ebensperger & Méndez, 2018). Además, dependiendo la extensión del muñón se denominan de tercio distal, medio o proximal.

2.5 Componentes hardware para medir señales mioeléctricas

Se refieren a los elementos físicos que constituyen un sistema computacional o dispositivo electrónico. Estos componentes desempeñan un papel esencial en el funcionamiento y operación de la tecnología, contribuyendo a su capacidad y rendimiento. Lo distintivo de estos elementos es su naturaleza tangible, lo que significa que son perceptibles a través de los sentidos, permitiendo su observación directa y manipulación física (Michaud, 2022).

2.5.1 Tarjetas de desarrollo

Son placas electrónicas diseñadas con el propósito de diseñar, desarrollar y prototipar proyectos relacionados con la electrónica y los sistemas embebidos. Estas placas integran elementos clave, como microcontroladores, interfaces de entrada y salida, y con frecuencia incluyen recursos de software y documentación de apoyo, simplificando la creación y la experimentación en el ámbito de las aplicaciones electrónicas e informáticas (Jolles, 2021).

Arduino

Arduino es una plataforma de código abierto que combina hardware y software para la creación de proyectos interactivos, tanto para entusiastas como para profesionales de la electrónica. La plataforma se basa en una placa de desarrollo que incorpora un microcontrolador y se acompaña de un entorno de programación que permite su programación y su carga del código (Bravo, 2018). La familia Arduino abarca diversos modelos y placas, cada uno con características particulares diseñadas para atender una variedad de aplicaciones y requerimientos de desarrollo. Los microcontroladores Arduino tienen entradas y salidas que se pueden utilizar para obtener información y en base a los datos recibidos enviar la salida correspondiente.(Shakirovich Ismailov, 2022). En la Ilustración 2-14 se representa el IDE de Arduino.

Ilustración 2-14 Arduino IDE

Fuente: (Tene & Tene, 2022)

Raspberry Pi

El Raspberry Pi que se observa en Ilustración 2-15 se compone de computadoras de placa única que destacan por su bajo costo, tamaño compacto y alto rendimiento (Jolles, 2021). Estas computadoras tienen aplicaciones versátiles, desde la educación hasta proyectos de bricolaje, la "familia" Raspberry Pi abarca una variedad de modelos y versiones, cada uno con características particulares diseñadas para satisfacer distintas demandas y propósitos.

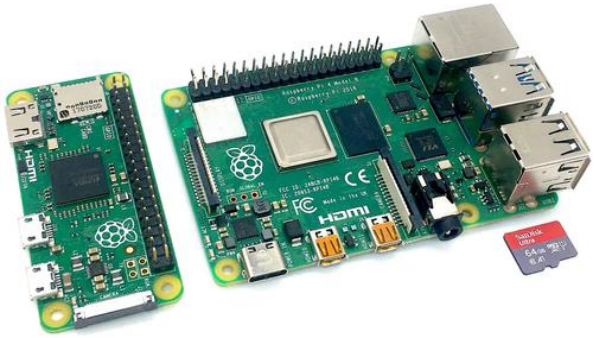


Ilustración 2-15 Raspberry pi

Fuente: (Jolles, 2021)

ESP

Se refiere a una gama de microcontroladores y módulos de conectividad inalámbrica desarrollada por Espressif Systems. Estos elementos se han concebido específicamente para posibilitar la integración de capacidades de Wifi y Bluetooth en proyectos orientados al Internet de las Cosas (IOT) y sistemas embebidos. En la familia ESP (Ilustración 2-16) destacan microcontroladores populares como el ESP8266 y el ESP32 que incorpora Wi-Fi y Bluetooth con una velocidad de hasta 150Mbps, un ADC (convertidor análogo-digital) de 10 pines, módulo de gestión de energía, receptor amplificador receptor, seguridad, filtrado, etc (Sandeep Rao et al., 2021).

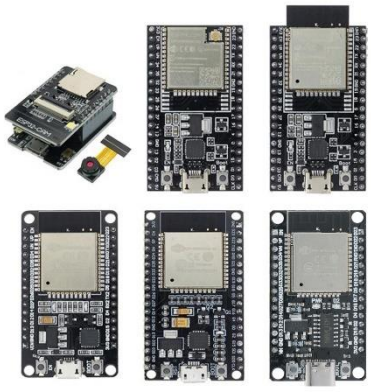


Ilustración 2-16 Diferentes tipos de ESP

Fuente: https://n9.cl/y7bfm

En la Tabla 2-5 se describen las características de las diferentes tarjetas de desarrollo.

Tabla 2-5 Resumen de características entre tarjetas de desarrollo

Característica	Arduino Uno	Arduino Nano	Raspberry Pi Pico	ESP32	ESP8266
Microcontrolador	ATmega328P	ATmega328P	RP2040	Xtensa LX6	Xtensa LX106
Velocidad de CPU	16 MHz	16 MHz	133 MHz	160 MHz	80 MHz
Memoria Flash	32 KB	32 KB	2 MB	4 MB	4 MB
RAM	2 KB	2 KB	264 KB	520 KB	80 KB
GPIO	14	14	26	36	17
Interfaces de Comunicación	UART, I2C, SPI	UART, I2C, SPI	UART, I2C, SPI, PWM	UART, I2C, SPI, PWM	UART, I2C, SPI, PWM
Wi-Fi	No	No	No	Sí	Sí
Bluetooth	No	No	No	Sí	No
Alimentación	5V	5V	3.3V	3.3V	3.3V
Software de desarrollo	Arduino IDE	Arduino IDE	C/C++, Micro Python, ESP-IDF, Platform IO	Arduino IDE, ESP-IDF, Platform IO	Arduino IDE, ESP8266 Arduino Core
Compatibilidad	Amplia	Amplia	Limitada	Amplia	Amplia
Interfaces adicionales	-	-	2 × UART, 2 × I2C, 2 × SPI, 3 × PWM	CAN, Ethernet,	
Precio aproximado	$20 - $25	$10 - $15	$4 - $5	$5 - $10	$3 - $5

Realizado por: Vaca-Paguay, 2024.

2.5.2 Electrodos de superficie

Estos electrodos (Ilustración 2-17) se colocan sobre la piel, en el músculo de interés para registrar las señales eléctricas generadas por la actividad muscular.

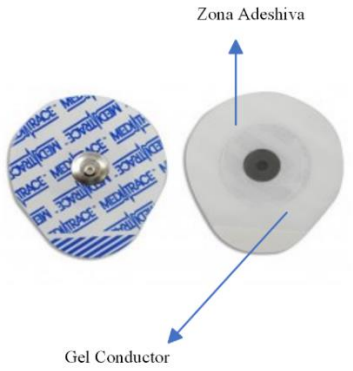


Ilustración 2-17 Electrodos de superficie

Fuente: Vaca-Paguay, 2023

2.5.3 Electrodos de aguja

Estos electrodos son invasivos que se insertan directamente en el músculo. Estos permiten mediciones más precisas de la actividad muscular (Suberviola, 2019). En la Ilustración 2-18 se pueden observar estos electrodos.

Ilustración 2-18 Electrodos de aguja

Fuente: https://www.micromed.com/es-ES/productos/electrodos/

2.5.4 Sensores

Son dispositivos que pueden ser tanto electrónicos como mecánicos, que permiten identificar y cuantificar los cambios de las condiciones físicas o del entorno (Azpilcueta, 2020). Estos sensores desempeñan la función de transformar las variaciones físicas en señales eléctricas, que posteriormente pueden ser sometidas a procesamiento y análisis. El resultado es la capacidad de supervisar y regular sistemas, así como tomar decisiones dentro de una amplia diversidad de aplicaciones, que van desde el ámbito de la automatización industrial hasta la creación de productos de consumo y dispositivos médicos.(García, 2020). En la Ilustración 2-19 se pueden ver los diferentes tipos de sensores que son comunes en el mercado.

Ilustración 2-19 Tipos de sensores comunes

Fuente:https://electropreguntas.com/los-diferentes-tipos-de-sensores-magneticos-para-tu-proyecto/

2.5.5 Sensores utilizados para medir señales mioeléctricas o EMG

Son dispositivos diseñados para adquirir y registrar la actividad eléctrica generada por la actividad muscular en el cuerpo humano. Estos sensores detectan las variaciones eléctricas en la piel que resultan de la contracción de los músculos y las convierten en señales eléctricas normalmente en el orden de los milivoltios que pueden ser analizadas por una tarjeta de desarrollo. Los sensores de señales mioeléctricas son esenciales en campos como la electromiografía (EMG). (Suberviola, 2019). En la Ilustración 2-20 se puede ver un tipo de sensor para medir señales mioeléctricas.

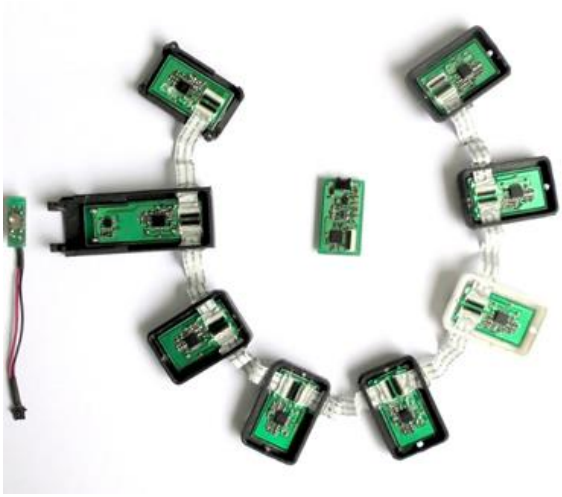


Ilustración 2-20 Brazalete de sensores (EMG)

Fuente: https://investigacion.pucp.edu.pe/grupos/girab/proyecto/brazalete-de-sensores-emg/

2.5.6 Sensores de impulsos nerviosos

Estos sensores se utilizan para medir la actividad eléctrica en los nervios y son relevantes en aplicaciones de diagnóstico neuromuscular y neurofisiología. Como se observa en la Ilustración 2-21.

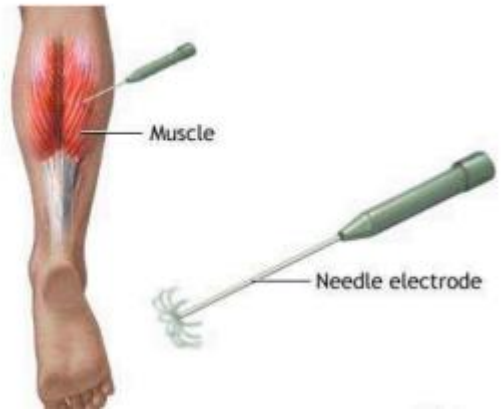


Ilustración 2-21 Sensor de EMG de aguja

Fuente: (Suberviola, 2019).

2.5.7 Sensores de fuerza muscular y presión

No miden señales directamente, sino que se utilizan para complementar la información obtenida a partir de señales mioeléctricas al medir la fuerza muscular y la presión ejercida por los músculos durante el desarrollo de una actividad. En la Ilustración 2-22 se puede ver un sensor celda de carga que sirve para medir la fuerza muscular.

Ilustración 2-22 Celda de carga

Fuente: https://equipesajes.com/inicio/74-celda-de-carga-tipo-viga-.html

En la Tabla 2-6 se puede comprender la comparativa de diferentes sensores que pueden ser utilizados para medir la actividad muscular.

Tabla 2-6 Comparativa entre tecnologías y modelos utilizados para medir la actividad muscular.

Tipo de Sensor	Modelo	Características Principales	Características Técnicas	Precio Aproximado
Sensores EMG	MyoWare	Detecta actividad muscular mediante electrodos.	Ganancia ajustable, rango de frecuencia, interfaz con microcontroladores.	$50 - $100
	Delsys Trigno	Sistema inalámbrico para medición de EMG intramuscular.	Canales de registro, precisión, software de análisis.	$1500 - $3000
Sensores de Impulsos Nerviosos	Electromiografía de Agujas	Mide actividad eléctrica dentro de los músculos.	Profundidad de penetración, número de canales.	Variables según tipo
	Potenciales Evocados	Registra actividad nerviosa en respuesta a estímulos.	Tiempo de respuesta, tipos de estímulos medibles.	Variables según tipo
Sensores de Fuerza Muscular y Presión	Celdas de Carga	Mide la fuerza ejercida por los músculos.	Rango de medición, precisión, tamaño y forma.	$50 - $500
	Sensores de Presión Piezoeléctricos	Detecta presión aplicada en áreas específicas.	Sensibilidad, área de detección, resistencia a la corrosión.	$20 - $200

Realizado por: Vaca-Paguay, 2024.

2.6 Software para adquisición y procesamiento de señales

El procesamiento y análisis de señales en la adquisición de datos constituye un pilar fundamental en campos como la medicina, la ingeniería y la investigación científica. Para llevar a cabo esta tarea de manera efectiva, se requiere el empleo de softwares especializados que sean capaces de adquirir, interpretar y procesar con precisión la información contenida en las señales. Estos softwares se convierten en herramientas esenciales para entender y extraer conclusiones

significativas a partir de datos complejos. (Vasconcellos, 2022). En la Ilustración 2-23 se observa un ejemplo de una interfaz de usuario para la adquisición de datos.

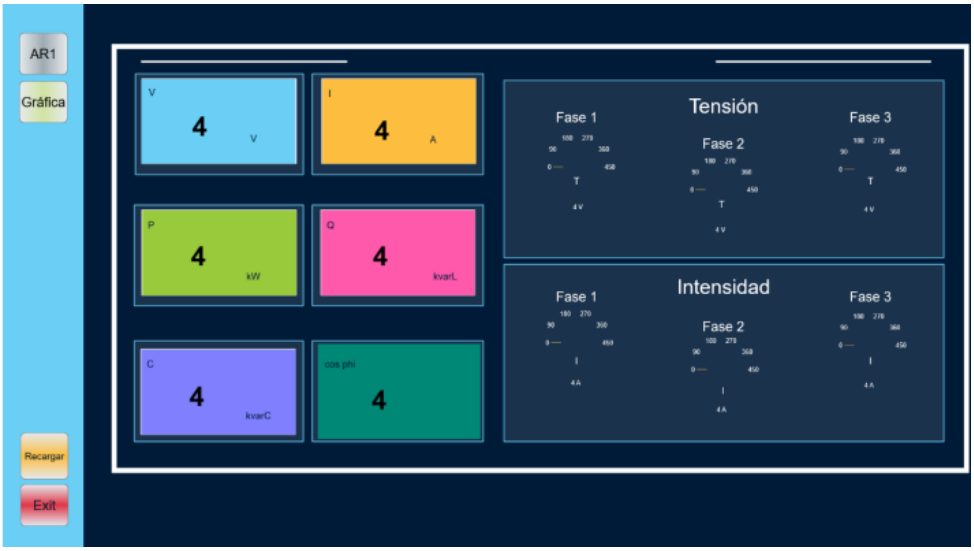


Ilustración 2-23 Interfaz de usuario sistema adquisición datos de consumo eléctrico

Fuentes:(Pagán González, 2023)

2.6.1 Lenguaje C

Reconocido por su velocidad y eficiencia, especialmente en sistemas integrados. Su enfoque se centra en un control más directo del hardware. Aunque requiere una sintaxis más precisa y detallada en comparación con otros lenguajes de programación de más alto nivel actuales, su capacidad para operar de manera cercana al hardware lo hace especialmente valioso en aplicaciones donde se busca un rendimiento óptimo. La necesidad de predefinir variables y la compilación previa a la ejecución son aspectos que pueden resultar desafiantes, pero su control preciso y su eficiencia en el manejo de recursos son ventajas clave en entornos donde la velocidad y la optimización son prioritarias. (Vasconcellos, 2022).

2.6.2 Matlab

Inicialmente concebido como una herramienta interactiva para cálculos matriciales, este software evolucionó hasta convertirse en un lenguaje de programación de alto nivel. A pesar de su ritmo comparativamente más lento que el de C, su gran ventaja reside en su capacidad para interpretar el código dinámicamente durante la ejecución. Está basado en Java pero simplifica notablemente la escritura, depuración y desarrollo de programas. Este software destaca por su habilidad en cálculos basados en matrices y su eficiencia en el procesamiento de imágenes. (Moler & Little, 2020)

2.6.3 Python

En (Sadriddinovich Jalolov, 2023), se indica que Python es un lenguaje de programación potente y versátil ampliamente utilizado para resolver problemas complejos en diversos campos. Este lenguaje de programación, caracterizado por su enfoque orientado a objetos y su condición de código abierto, se posiciona como una herramienta de alto nivel en el ámbito de la programación. Su estructura flexible y clara permite una escritura de código más intuitiva y adaptable.

Su capacidad para manejar eficientemente procesos en paralelo resulta crucial en entornos donde la multitarea es fundamental. Además, cuenta con una comunidad de desarrolladores activa y diversa que contribuye constantemente al desarrollo de nuevas herramientas y librerías. (Vasconcellos, 2022). El acceso libre y abierto a una amplia gama de librerías es uno de los puntos fuertes de este lenguaje, lo que facilita significativamente el desarrollo de proyectos complejos. Un aspecto especialmente destacable es su enfoque cada vez más prominente en el campo del aprendizaje automático y profundo. En la Tabla 2-7 se encuentran las ventajas y desventajas de los diferentes software para la adquisición y procesamiento de señales.

Tabla 2-7 Ventajas y desventajas entre los diferentes softwares para adquisición y procesamiento de señales.

Software	Ventajas	Desventajas	Características
Lenguaje C	Alta velocidad de ejecución.	Requiere más líneas de código que otros lenguajes.	Acceso de bajo nivel al hardware.
	Control directo sobre la manipulación de memoria.	Curva de aprendizaje empinada para principiantes.	Optimizado para operaciones de bajo nivel.
	Eficiencia y rendimiento en tiempo real.	Menos portabilidad entre sistemas operativos.	Buena capacidad para sistemas embebidos.
MATLAB	Potentes herramientas de procesamiento de señales.	Licencias costosas para uso comercial.	Entorno de desarrollo con GUI para análisis.

	Facilidad para visualización y análisis de datos.	Menor flexibilidad para programación general.	Extensa colección de toolbox para señales.
	Amplia variedad de funciones predefinidas.	No es un lenguaje de programación convencional.	Integración con hardware mediante toolboxes.
Python	Potentes herramientas de procesamiento de señales.	Licencias costosas para uso comercial.	Entorno de desarrollo con GUI para análisis.
	Facilidad para visualización y análisis de datos.	Menor flexibilidad para programación general.	Extensa colección de toolbox para señales.
	Amplia variedad de funciones predefinidas.	No es un lenguaje de programación convencional.	Integración con hardware mediante toolboxes.

Realizado por: Vaca-Paguay, 2024.

2.7 Inteligencia artificial

La definición más extendida de inteligencia artificial hace referencia a una serie de algoritmos diseñados para ejecutar tareas que normalmente se atribuyen a la inteligencia humana. (Cobo & LLoret Iglesias, 2023). Por otro lado, (Ramírez & Ramírez, 2023) dice que la IA es un área de la informática que tiene como objetivo hacer que las máquinas hagan cosas inteligentes, es decir, aprendan y resuelvan problemas de forma similar a la inteligencia natural de los humanos y los animales. En IA, un agente inteligente recibe información del entorno, realiza cálculos para decidir qué acción tomar de forma autónoma para lograr un objetivo. La IA puede mejorar su rendimiento con el entrenamiento sobre un conjunto de datos denominado aprendizaje. Y por último (Rozo-García, 2020) asegura que es una rama del conocimiento de naturaleza multidisciplinar que involucra campos dentro de las ciencias de la computación, la información, la lógica, la matemática, la estadística, la biología, la psicología, la filosofía, la lingüística y otras áreas.

2.7.1 Machine Learning

El Machine Learning (ML) o aprendizaje automático es una parte de la IA que busca desarrollar algoritmos con la capacidad de generalizar comportamientos y aprender patrones a partir de un conjunto de datos determinado (Del Carmen & Ramos, 2021). De este modo son capaces de resolver problemas sin la necesidad de haber sido programados explícitamente para ello. Todo proceso de ML cuenta con una fase de entrenamiento, ajuste, predicción y evaluación. A partir de un gran número de ejemplos, se elabora un modelo que puede deducir y generalizar un comportamiento tanto para información conocida como para aquella que el sistema no conoce.

(Quevedo & Gómez Donoso, 2021). El Machine Learning tiene diferentes paradigmas como se muestra en la Ilustración 2-24.

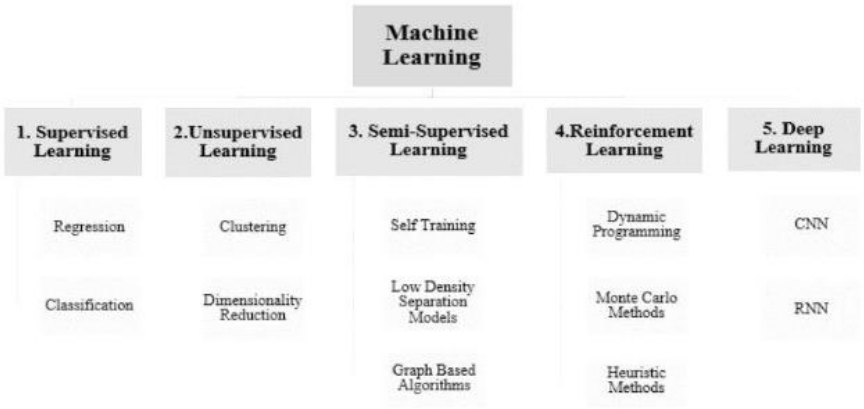


Ilustración 2-24 Tipos de Machine Learning

Fuente: (Salamanca & Castro, 2020)

2.7.2 Deep Learning

Es una parte del ML que se relaciona con el aprendizaje profundo. Esto depende de la cantidad de niveles de neuronas que se estructuran en una Red Neuronal Artificial (ANN, por sus siglas en inglés) o de los tipos de Redes Neuronales enfocadas en la extracción automática de patrones en los datos (Salamanca & Castro, 2020).

Las capas de las ANN se organizan en tres categorías fundamentales: la Capa de Entrada, responsable de recibir los datos a procesar; la Capa Oculta, destinada a procesar la información de manera interna de forma no lineal; y la Capa de Salida, encargada de proporcionar el resultado del procesamiento

En una ANN de profundidad considerable se pueden encontrar múltiples capas ocultas como se visualiza en la Ilustración 2-25, que le permite afrontar tareas de mayor complejidad al realizar transformaciones y cálculos más sofisticados en los datos. Por el contrario, una ANN más superficial consta únicamente de una capa oculta, limitando su capacidad para abordar tareas complejas al realizar un procesamiento más simple y directo de la información recibida (Noailles, 2022). Este incremento en la profundidad de la red conlleva la capacidad de aprender y representar características cada vez más abstractas y complejas en los datos de entrada, ampliando su potencial para resolver problemas desafiantes en diversas áreas. (Hernández Montejano, 2022).

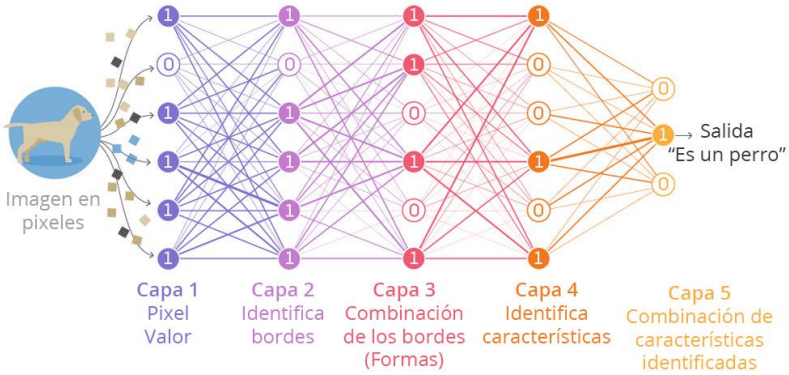


Ilustración 2-25 Ejemplo de Deep Learning

Fuente: https://www.quantamagazine.org

2.7.3 Tipos de aprendizaje automático.

2.7.3.1 Aprendizaje supervisado

Los conjuntos de datos de salida para este tipo de aprendizaje están etiquetados previamente. Por tanto, las variables de salida son clases o categorías. A través de esto, el modelo puede ser capaz de realizar predicciones de ejemplos no etiquetados con alta precisión. Principalmente, se asocia con problemas de clasificación, regresión y ranking problema. (Hermitaño, 2022).

2.7.3.2 Aprendizaje no supervisado

El aprendizaje no supervisado utiliza datos no etiquetados durante su entrenamiento. Estos algoritmos se utilizan principalmente en tareas en las que se deben analizar datos para extraer patrones de manera automática, relaciones en los datos, entre otras. Se utilizan principalmente en reconocimiento de patrones y agrupamiento de datos. (Padilla & Nicolalde, 2022)

2.7.3.3 Aprendizaje semisupervisado:

En el aprendizaje semisupervisado, se utilizan tanto datos etiquetados como datos no etiquetados. Esto es particularmente útil cuando se tiene gran cantidad de datos de entrada y pocos de salida (Brusil, 2020). El procedimiento básico es agrupar los datos en diferentes grupos usando un algoritmo de aprendizaje no supervisado y luego usar los datos etiquetados existentes para etiquetar el resto de los datos. Los algoritmos más populares incluyen el autoaprendizaje, los métodos generativos, los modelos mixtos y los métodos basados en gráficos. El aprendizaje semisupervisado generalmente se usa en análisis de voz, clasificación de contenido de Internet y clasificación de secuencias de proteínas (Ramírez & Ramírez, 2023)

2.7.3.4 Aprendizaje por refuerzo

Permite la generación de un agente capaz de aprender qué hacer o qué acciones tomar para maximizar una función de recompensa. El agente que aprende no recibe explícitamente las acciones que debe tomar, sino que debe ir descubriendo, en general, usando el método de prueba y error.(Quevedo & Gómez Donoso, 2021).

En la Tabla 2-8 se puede observar la comparativa de los diferentes tipos de ML.

Tabla 2-8 Comparativa de tipos de Machine Learning

Aprendizaje/ Característica	Complejidad del modelo	Entrenamiento	Ventajas	Desventajas	Ejemplos
Supervisado	Puede ser complejo, dependiendo de la tarea	Requiere un conjunto de datos etiquetados	Precisión mejorada, ampliamente aplicable	Necesidad de datos etiquetados, dependencia de calidad de etiquetas	Clasificación de correos electrónicos como spam o no spam
No supervisado	Puede ser menos complejo que el supervisado	Puede funcionar con datos no etiquetados	Descubrimiento de patrones intrínsecos, exploratorio	Falta de interpretación directa, dependencia de inicialización	Agrupamiento de noticias en temas similares
Semisupervisado	La complejidad del modelo puede variar dependiendo de la cantidad de datos etiquetados y no etiquetados	Combina datos etiquetados y no etiquetados	Eficiencia en recopilación de datos, mayor precisión	Dependencia de calidad de etiquetas parciales, complejidad adicional	Clasificación de imágenes con etiquetas parciales
Por refuerzo	Puede ser complejo debido a la necesidad de tomar decisiones secuenciales	Se realiza a través de la interacción continua con un entorno, no utiliza etiquetas	Adaptabilidad, aprendizaje continuo	Proceso de enseñanza, inestabilidad en algunos escenarios	Aprendizaje de un agente de juegos para maximizar la puntuación

Realizado por: Vaca-Paguay, 2024.

2.7.4 Técnicas de Machine Learning

2.7.4.1 Regresión lineal y logística

Se utiliza para modelar la relación entre una variable dependiente continua y una o más variables independientes, asumiendo que tienen una relación lineal. Su objetivo es predecir un valor

numérico. La regresión lineal comprende una línea recta que divide los puntos de datos en un diagrama de dispersión con la menor distancia posible de cada uno de ellos. (Aliaga, 2022)

Por otro lado, la logística es aplicable para problemas de categorización. Estima los valores reales y que la probabilidad de que una observación pertenezca a una clase especifica (Alba & Calle, 2020).

Estos modelos, aunque útiles para patrones simples, pueden no ser efectivos para patrones más complejos. En la Ilustración 2-26 se puede ver un ejemplo de regresión lineal.

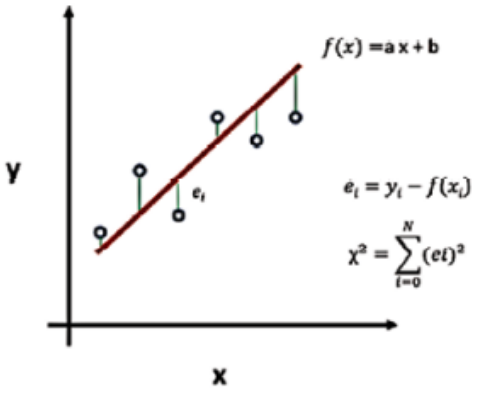


Ilustración 2-26 Regresión lineal

Fuente: (Ramírez & Ramírez, 2023)

2.7.4.2 Arboles de decisión

En (Ramírez & Ramírez, 2023) se indica que un árbol de decisión es uno de los métodos de aprendizaje supervisado no paramétricos ampliamente utilizados, ya que, se pueden utilizar para problemas de clasificación y regresión. Un conjunto de reglas se puede derivar de un árbol de decisión.

En la Ilustración 2-27 se representa un ejemplo de árbol de decisión.

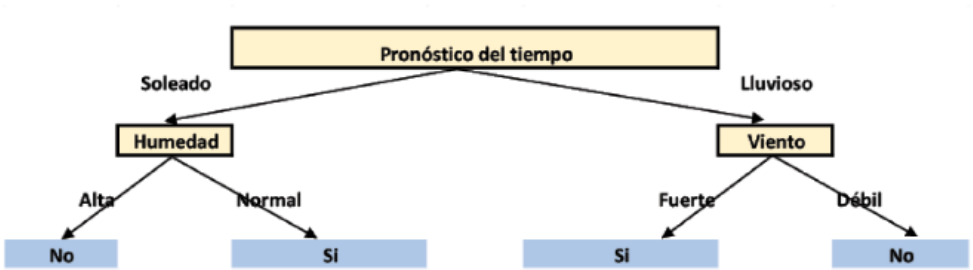


Ilustración 2-27 Ejemplo de árbol de decisión

Fuente (Ramírez & Ramírez, 2023)

2.7.4.3 Random forest

Random forest o bosque aleatorio es un algoritmo que utiliza múltiples árboles de decisión como se observa en la Ilustración 2-28. Un solo árbol de decisión puede ser insuficiente para algunas aplicaciones. En este algoritmo se crea aleatoriamente un conjunto de árboles de decisión donde cada árbol trabaja en un subconjunto aleatorio de muestras extraídos de los datos (Pérez, 2019).

Un bosque aleatorio combina las salidas preliminares de los árboles de decisión individuales para generar una salida final. Un bosque aleatorio es un algoritmo de aprendizaje conjunto que se puede utilizar para problemas de clasificación y de regresión con enfoque supervisado o no supervisado. (Ramírez & Ramírez, 2023).

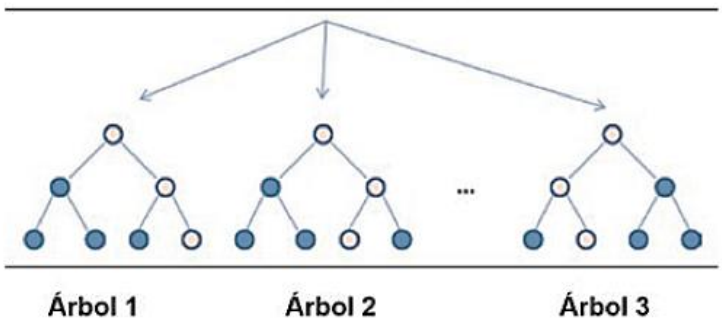


Ilustración 2-28 Random forest

Fuente: (Ramírez & Ramírez, 2023)

2.7.4.4 Redes neuronales

Redes neuronales artificiales (ANN)

Son modelos computacionales basados en un modelo biológico específicamente de la actividad bioeléctrica de las neuronas en el cerebro. (Bodero et al., 2019). Están compuestas por capas de neuronas interconectadas que procesan datos mediante entradas, capas ocultas que realizan cálculos de tipo matemático y una capa de salida que produce resultados. (Aljure Jiménez et al., 2021). Durante el entrenamiento, ajustan sus conexiones para aprender patrones, siendo útiles en áreas como reconocimiento, predicción y análisis de datos en campos diversos. El surgimiento del aprendizaje profundo ha impulsado redes más complejas, permitiéndoles manejar tareas complejas y conjuntos de datos extensos.

La Ecuación 1 describe el proceso de una neurona artificial en una capa de una ANN completamente conectada:

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b \quad (1)$$

$$a = f(z)$$

Donde, x_1,x_2, …,x_n son las entradas, w_1,w_2, …,w_n son los pesos correspondientes a las entradas, *b* es el sesgo (bias), *z* es la suma ponderada considerando el sesgo, *f* (z) es la función de activación y *a* la variable de salida.

Redes neuronales convolucionales (CNN)

Comparten semejanzas con las redes neuronales tradicionales. Cada unidad neuronal, con sus pesos aprendidos de los datos, recibe entradas y lleva a cabo operaciones de producto escalar. Asimismo, emplean una función de pérdida con relación a la capa precedente y pueden utilizar una función no lineal. Según (Aljure Jiménez et al., 2021) la distinción principal entre las CNN y las ANN radica en que las primeras suelen recibir imágenes como entrada, lo cual posibilita la incorporación de características específicas en la red para reducir la cantidad de parámetros necesarios.

Para las Redes Neuronales Convolucionales (CNN), se utilizan operaciones de convolución en las capas convolucionales seguidas por funciones de activación.

En una capa convolucional se define mediante la Ecuación 2:

$$z = (w * x) + b \qquad (2)$$

$$a = f(z)$$

Donde, x es la entrada, w es el kernel o filtro convolucional, *b* es el sesgo (baias), *z* es el resultado de la convolución más el sesgo, *f* () es la función de activación y *a* la variable de salida.

En la Ilustración 2-29 se muestra como es el proceso de una red convolucional.

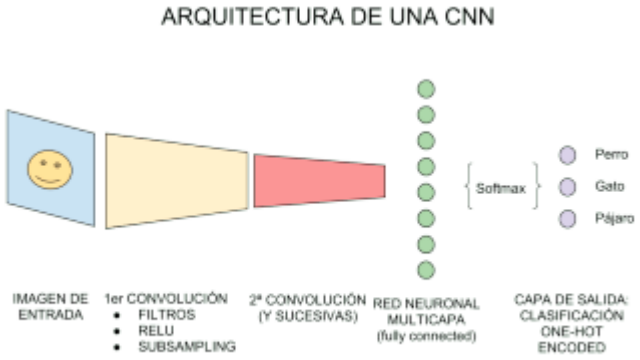


Ilustración 2-29 Red neuronal Convolucional (CNN)

Fuente: (Bonilla, 2020)

En la Tabla 2-9 se realiza una comparación entre las técnicas de IA que existen.

Tabla 2-9 Comparativa entre técnicas de IA

Modelo/ Característica	Complejidad del modelo	Manejo de características	Rendimiento de problemas	Tiempo de entrenamiento	Interpretación del modelo
Regresión lineal	Baja, sencillo	Requiere normalización	Adecuado para relaciones lineales	Rápido	Relativamente alta
Regresión logística	Baja, sencillo	Requiere normalización	Adecuado para clasificación	Rápido	Relativamente alta
Árboles de decisión	Baja, sencillo	Maneja datos mixtos	Eficiente en datos estructurados	Rápido	Alta, sigue ramas para entender las decisiones del modelo
Random forest	Media, mejorando la generalización y reduce el sobreajuste	Maneja datos mixtos	Buen rendimiento en problemas	Puede ser largo en conjuntos grandes	Media, visibilidad sobre las características más importantes
Redes neuronales	Alta, aprende patrones complejos, puede sufrir de sobreajuste en conjuntos de datos pequeños	Requiere normalización	Buen rendimiento en problemas	Puede ser largo en conjuntos grandes	Baja, debido a su complejidad

Fuente:(Barahona & Jaramillo, 2022)

2.7.5 Inteligencia artificial aplicado al modelado de señales

Utiliza algoritmos y metodologías de inteligencia artificial para examinar, comprender y representar una variedad de señales en distintos contextos de aplicación. Estas señales pueden abarcar desde datos biomédicos, señales mioeléctricas, señales de audio, imágenes, secuencias temporales entre otras.

La inteligencia artificial se emplea para crear modelos con la capacidad de identificar patrones, efectuar proyecciones, extraer información pertinente y tomar decisiones basadas en los datos obtenidos a partir de estas señales. Este logro se materializa a través de enfoques como el aprendizaje automático, las redes neuronales, el procesamiento de señales digitales y la optimización (Solanet, 2021).

Además, la inteligencia artificial desempeña un papel fundamental en la identificación de patrones de la actividad muscular, lo cual es esencial en una diversidad de aplicaciones que van desde la rehabilitación hasta la interfaz cerebro-computador y la biomecánica. Esta integración

mejora la precisión y la eficiencia en la interpretación y el procesamiento de datos, lo que genera un impacto considerable en la industria y academia.

En la Tabla 2-10 se visualizan las ventajas y desventajas que se tiene al modelar las señales con Machine Learning.

Tabla 2-10 Ventajas y desventajas de modelar señales con Machine Learning

Característica	Adaptabilidad a diferentes señales	Manejo de datos ruidosos y no lineales	Adaptabilidad a cambios en el tiempo	Riesgo de sobreajuste
Ventajas	Versatilidad para adaptarse a diversos tipos de señales	Capacidad para manejar datos ruidosos y relaciones no lineales	Puede manejar información secuencial y adaptarse a cambios en el tiempo	Generalización a nuevos datos si se evita el sobreajuste
Desventajas	Necesidad de ajuste de hiperparámetros para diferentes tipos de señales	Riesgo de sobreajuste a datos ruidosos si no se maneja adecuadamente	Modelos más complejos pueden requerir más recursos computacionales	Riesgo de sobreajuste si no se gestiona adecuadamente

Realizado por: Vaca-Paguay, 2024.

CAPÍTULO III

3. MARCO METODOLÓGICO

3.1 Requerimientos

Este proyecto busca desarrollar un sistema capaz de adquirir señales mioeléctricas mediante sensores conectados a un sistema electrónico y a una aplicación desarrollada en Python para su procesamiento a través de un modelo avanzado de Machine Learning capaz de interpretar y predecir patrones precisos de activación muscular durante la ejecución de actividades previamente definidas.

3.1.1 Requerimientos de Hardware

Las especificaciones requeridas para el funcionamiento del hardware son las siguientes:

Sensores mioeléctricos: se utilizarán tres sensores, integrados en una única pulsera, cada uno con tres electrodos. Estos sensores se ubicarán en tres zonas distintas del brazo: la primera zona en el tercio proximal del antebrazo, por debajo del codo; la segunda en el tercio distal del brazo, región del brazo inferior; y la tercera en el tercio proximal del brazo, región anterior al hombro.

Dispositivo de Adquisición de datos: para la adquisición de datos, se utiliza una tarjeta de desarrollo, la cual debe tener una capacidad de muestreo de alta frecuencia. Esta tarjeta está integrada en una placa de circuito impreso (PCB) que cuenta con tres pines de salida, utilizados como fuente de alimentación, además de tres pines de tierra (GND) y tres pines para procesamiento de las señales provenientes de los sensores.

Computadora: la computadora debe cumplir especificaciones que garanticen un rendimiento óptimo durante la adquisición, procesamiento y análisis de señales mioeléctricas. Para esto debe contar con una memoria RAM de al menos 8GB, puertos USB para facilitar la conexión de dispositivos como los sensores mioeléctricos mediante la tarjeta de desarrollo.

3.1.2 Requerimientos de Software

- Se requiere de un lenguaje de programación de alto nivel como Python, para desarrollar una aplicación que permita la recolección, almacenamiento y organización de los datos provenientes de los sensores.
- Además, esta aplicación permitirá, procesar, visualizar y almacenar de manera precisa la información obtenida desde la tarjeta de desarrollo, asegurando una gestión eficiente y parametrizada de los datos recolectados.

- La aplicación deberá ofrecer diversas funcionalidades, permitiendo ajustes detallados como la configuración de puertos, la velocidad de transmisión de datos, los intervalos de adquisición de señal y los tiempos de espera específicos para cada movimiento y músculo involucrado.
- La aplicación deberá disponer de una interfaz gráfica que permita visualizar en tiempo real los datos muestreados de las señales mioeléctricas durante su adquisición. Posteriormente, permitirá revisar los datos almacenados, facilitando su análisis e investigación.
- Finalmente se precisa la implementación de algoritmos de aprendizaje automático adecuados para la predicción de las señales mioeléctricas, permitiendo la generación y entrenamiento de modelos que interpreten de manera eficiente esta información.

3.2 Concepción de la arquitectura

La concepción de la arquitectura se describe en la Ilustración 3-1, la cual, consta de 5 etapas: adquisición y procesamiento de señales, adquisición y visualización de datos, clasificación y normalización, entrenamiento y almacenamiento del modelo final de Machine Learning.

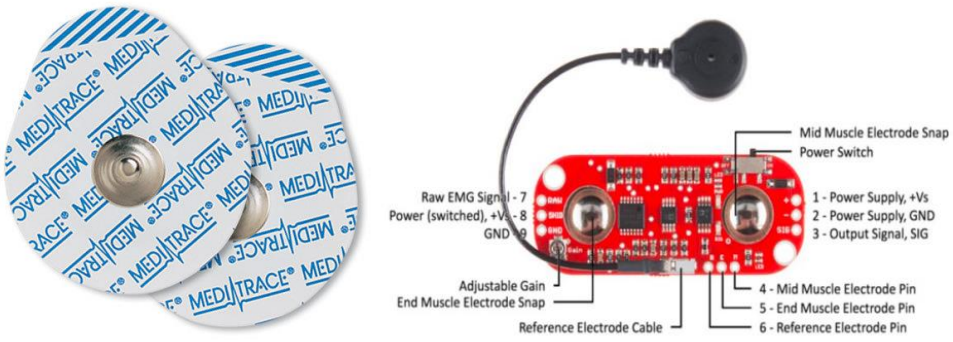
Ilustración 3-1 Concepción General de la Arquitectura

Realizado por: Vaca-Paguay, 2024.

3.2.1 Etapa de adquisición y procesamiento de señales

Para la adquisición de señales se utilizan 3 sensores mioeléctricos MyoWare como se muestra en la Ilustración 3-2 (b). Estos sensores son de bajo costo y permiten capturar la actividad eléctrica de los músculos mediante la tecnología de electromiografía (EMG), a través de electrodos Kendall 230 Foam Medi-Trace (Ver Ilustración 3-2 (a)). Estos electrodos de espuma suave y flexible permiten una colocación cómoda y segura. Antes de ser colocados, es importante la limpieza de la piel, para proporcionar registros con bajos niveles de ruido. (Rodríguez, 2020) El dispositivo Myoware recibe las señales musculares, las amplifica con una la ganancia entre 0.01Ω y 100kΩ y luego las filtra antes de enviarlas al controlador. (Siddiq Ahmed et al., 2021). Cada sensor tiene un ajuste en ganancia con el siguiente detalle: *sensor 1*: ganancia de 39kΩ, *sensor 2*: de 50kΩ y *sensor 3*: de 51kΩ. La información detallada se muestra en el Anexo D.

Una vez configurados los sensores se coloca en una manilla de velcro para fijar y sujetar, asegurando una conexión estable y minimizando los posibles movimientos que puedan afectar la adquisición de la señal.



a) **b)**

Ilustración 3-2 Dispositivos usados para la adquisición y preprocesamiento de las señales musculares. **(a)** Electrodos Kendall 230 Foam Medi-Trace**, (b)** Sensor MyoWare

Fuente: (Siddiq Ahmed et al., 2021)

En la Tabla 3-1 se encuentra los parámetros del sensor mioeléctrico utilizado.

Tabla 3-1 Parámetros del sensor MyoWare

Parámetros	Mínimo	Típico	Máximo
Voltaje de suministro	+2.9V	+3.3V or +5V	+5.7V
Potenciómetro de ganancia ajustable	0.01 Ω	50 kΩ	100 kΩ
Voltaje de señal de salida	0V	-	+Vs
EMG EnvelopeRaw EMG (centered about +Vs/2)	0V	-	+Vs
Impedancia de entrada	-	110 GΩ	-
Corrientes de suministro	-	9 mA	14 mA
Relación de rechazo de modo común (CMRR)	-	110	-
Polarización de entrada	-	1 pA	-

Fuente:(*MyoWare-Muscle-Sensor-AT-04-001-*, 2015)

Para el procesamiento de señales se utiliza una tarjeta de desarrollo ESP-32 presentada en la Ilustración 3-3 y sus especificaciones en el Anexo E. Esta tarjeta se destaca por su capacidad para conversión de señales analógicas en digitales mediante sus pines ADC con una resolución de 12 bits que supera a otros microcontroladores. Su potencia, con un procesador dual-core de hasta 240 MHz, asegura el manejo eficiente de tareas complejas como la lectura de los canales analógicos y transmite los datos por UART, que es un protocolo para comunicación para envío de datos en forma serial (Jácome del Valle, 2022). Todo esto, junto con su accesibilidad y coste, la convierte en una opción sólida para proyectos que requieran lectura analógica precisa y transmisión de datos hacia una PC. Además, son MCU de bajo consumo energético, ya que, tiene el modo de bajo consumo deep sleep. (Ikiss, 2020) Algunas de las características relevantes para este trabajo se muestran en la Tabla 3-2.

Ilustración 3-3 Tarjeta de desarrollo ESP32S NodeMCU

Fuente: Lozano, 2021

Tabla 3-2 Especificaciones técnicas ESP32s

Especificación Técnica	Descripción
Entorno de desarrollo	Admite Arduino IDE, MicroPython y lenguaje nativo
Antena	Incorpora antena para conectividad inalámbrica
Procesador	Potente procesador dual-core Xtensa LX6 a 240 MHz
Consumo de energía	Eficiencia energética en estado de reposo
Conectividad	Wi-Fi 802.11 b/g/n, Bluetooth v4.2 BR/EDR
Voltaje de operación	Opera a 3.3 V
Protocolos de red	Compatibilidad con TCP/IP, UDP, HTTP, MQTT, WebSocket
Interfaces	Ofrece UART, SPI, I2C, GPIO y soporte para cámaras
Memoria	520 KB de SRAM, 4 MB de memoria flash
Velocidad del reloj	Hasta 80 MHz

Fuente: (Berrios Gómez & Rivera Herrera, 2022) y (Jácome del Valle, 2022)

Esta tarjeta de desarrollo tiene compatibilidad con las bibliotecas de Arduino (Ilustración 3-4) que simplifican la programación, y su amplia comunidad de desarrollo ofrece una gran cantidad de recursos y ejemplos en línea. (Nescolarde López, 2023). Esta plataforma permite controlar la adquisición, estandarización y transmisión de datos desde el ESP-32. Además, Arduino IDE ofrece una plataforma robusta para la configuración de los parámetros de comunicación UART asegurando una transmisión eficiente y sincronizada de datos en tiempo real, minimizando así la pérdida de información y latencia como se describe en la Tabla 3-3.

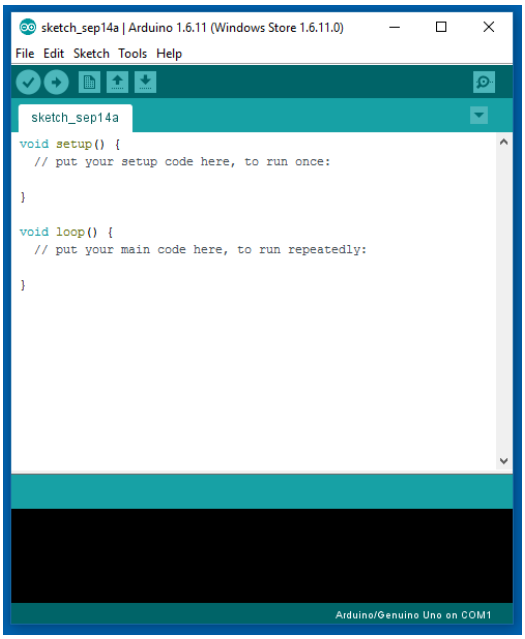


Ilustración 3-4 Arduino IDE
Fuente: https://apps.microsoft.com/detail/9nblggh4rsd8?hl=es-hn&gl=HN

Tabla 3-3 Especificaciones Arduino IDE

Especificación Técnica	Descripción
Facilidad de uso	Altamente amigable y accesible
Compatibilidad	Amplia compatibilidad con hardware, incluyendo ESP32
Control de UART	Permite un control detallado y específico de la comunicación UART
Documentación	Abundante y recursos disponibles en línea
Versatilidad	Ideal para aplicaciones enfocadas en hardware y sensores

Realizado por: Vaca-Paguay, 2024

En la Ilustración 3-5 se muestra cómo interactúan los componentes de hardware y software en la etapa de adquisición y procesamiento de datos de acuerdo, mediante el uso de un diagrama de bloques.

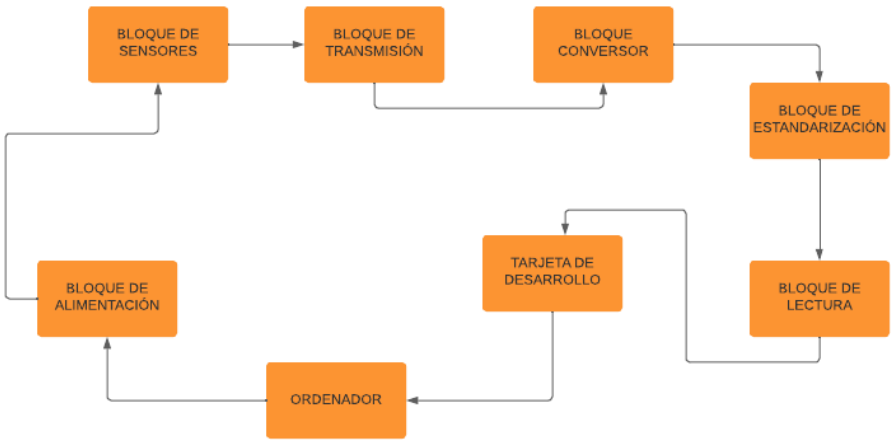


Ilustración 3-5 Diagrama de Bloques de la etapa adquisición y procesamiento de datos

Realizado por: Vaca-Paguay, 2024.

Ordenador: Para este bloque se utiliza un ordenador ya que por medio de este obtenemos la alimentación de la tarjeta de desarrollo, esta es quien procesa y almacena las señales digitalizadas provenientes de la tarjeta ESP-32.

Bloque de alimentación: Se obtienen a partir del ordenador para que pueda alimentar a la ESP-32 y a su vez a los sensores MyoWare obteniendo 3.3V en los pines de alimentación.

Bloque de sensores: Mediante la regulación de las ganancias de cada uno de los tres sensores MyoWare, se obtiene las señales analógicas por la contracción muscular que se produce al momento de realizar los 8 movimientos establecidos en la interfaz gráfica que son abrir y cerrar mano, levantar y bajar brazo, tomar objeto, levantar y bajar objeto y soltar objeto.

Bloque de transmisión de datos: Transmite los datos por medio de una comunicación serial con una velocidad de 115200 baudios.

Bloque conversor: Este bloque recibe los datos de los sensores a través de la comunicación ADC, su función principal es interpretar y convertir los datos en un formato entendible para la tarjeta de desarrollo.

Bloque de estandarización: En este bloque se busca ajustar la amplitud de las señales para que todas estén en un rango especifico de 0 a 5V, facilitando su procesamiento y análisis.

Bloque de lectura: El bloque de lectura recibe los datos de actividad mioeléctrica muscular proporcionados por los sensores MyoWare, con un tiempo de espera de lectura entre sensores de 100 milisegundos.

Tarjeta de desarrollo: En este bloque se utiliza la tarjeta ESP-32, la cual recibe la señal mioeléctrica proveniente de los sensores y esta a su vez la procesa y ordena para poder ser enviada al computador.

3.2.2 Etapa de adquisición y visualización de datos

PyCharm es un IDE de Python que implementa una funcionalidad híbrida para la generación y recuperación de código. (Xu et al., 2022) Este software sobresale en el desarrollo de programas debido a su conjunto robusto de herramientas y librerías de código abierto disponibles para Python. Como entorno integrado de desarrollo, simplifica la creación de interfaces gráficas al brindar soporte para bibliotecas como PySimpleGui, Tkinter, PyQt o Kivy. Según (Rathika et al., 2021) permite un diseño intuitivo y la implementación de interfaces interactivas. Esto permite controlar parámetros como velocidad de comunicación, tiempo de adquisición, selección de músculos y visualización de señales en tiempo real. Además, su compatibilidad con diversas frameworks y bibliotecas asegura la flexibilidad necesaria para desarrollar programas robustos para adquisición y procesamiento de señales. Estas características se muestran en la Tabla 3-4.

Tabla 3-4 Especificaciones del software PyCharm

Especificación	Descripción
Tipo de software	Entorno integrado de desarrollo (IDE) para Python
Creación de interfaces	Permite el diseño y desarrollo de interfaces gráficas para controlar parámetros y visualizar datos
Funcionalidades clave	Depurador potente, finalización de código inteligente, refactorización avanzada
Compatibilidad	Amplia compatibilidad con frameworks y bibliotecas de Python
Entorno de ejecución	Puede ser utilizado con entornos Python estándar o entornos basados en Anaconda
Facilidad de desarrollo	Interfaz amigable que simplifica el desarrollo y la depuración de aplicaciones complejas

Realizado por: Vaca-Paguay, 2024.

En la Ilustración 3-6 se presenta un diagrama de bloques que detalla la etapa de caracterización y visualización de datos.

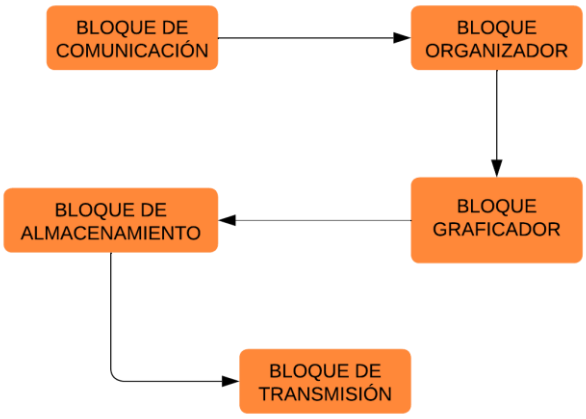


Ilustración 3-6 Diagrama de bloques etapa de caracterización y visualización de datos.

Realizado por: Vaca-Paguay, 2024.

Bloque de comunicación: Se encarga de establecer la comunicación con el microcontrolador a través del puerto COM Serial del PC. Lee los datos transmitidos por el microcontrolador y los prepara para su procesamiento.

Bloque organizador: Una vez que los datos han sido convertidos, el bloque organizador se encarga de organizarlos adecuadamente para facilitar su análisis.

Bloque graficador: Prepara los datos organizados para su representación gráfica que permitan entender de forma más clara y precisa la información obtenida.

Bloque de almacenamiento: Después de que los datos han sido procesados y visualizados, este bloque se encarga de guardar los resultados en un formato de almacenamiento adecuado, como archivos .csv

Bloque de transmisión: Transfiere los archivos procesados a un almacenamiento en la nube para facilitar el acceso y la colaboración remota con los datos.

3.2.2.1 Diseño e implementación de una interfaz gráfica para control, adquisición y visualización de datos.

La interfaz permite adquirir las señales mioeléctricas a través de los sensores MyoWare en tiempo real de los 8 movimientos ejecutados por las personas de prueba. Además, se visualizan las señales en tres canales y, finalmente en esta etapa se guardan todos estos datos en un directorio con archivos .csv.

El diseño de la interfaz y su funcionamiento se muestran en la Ilustración 3-7, donde se puede observar todos los parámetros que se debe configurar para la adquisición y visualización de datos. Esta interfaz de divide por secciones las cuales se describirán a continuación.

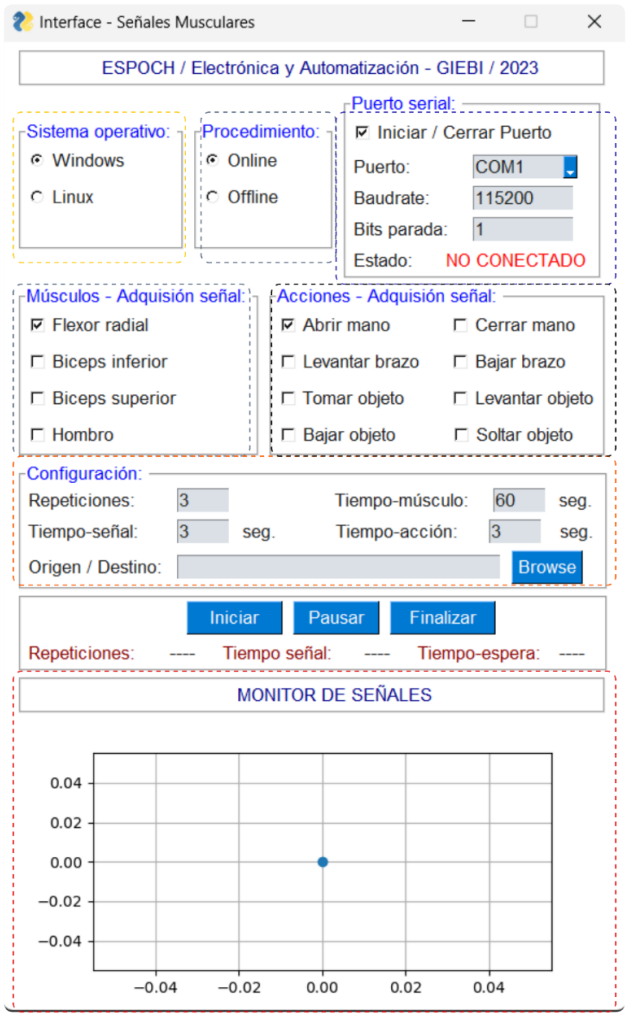


Ilustración 3-7 Interfaz gráfica desarrollada modo DESCONECTADO

Realizado por: Vaca-Paguay, 2024.

3.2.2.2 Parámetros de la interfaz

Sistema operativo: En esta sección se puede seleccionar dos tipos de sistema operativos con el que se puede trabajar, Windows o Linux. En este proyecto se ha utilizado Windows para realizar la adquisición de datos.

Puerto serial: En primer lugar, se puede iniciar o cerrar el puerto. Además, se tiene la selección del puerto por el cual está conectado la ESP-32, para este caso COM9. el Baudrate que es la velocidad de transmisión sde 115200 bits por segundo, los bits de parada en1 y por último el estado CONECTADO/DESCONECTADO del puerto. Una vez seleccionado el puerto correcto el estado cambiará a CONECTADO.

Músculos – Adquisición de señal: En esta sección se encuentran las 3 secciones de los músculos en donde se van a colocar los tres sensores para adquirir la señal.

Acciones – Adquisición de señal: Se pueden visualizar los 8 movimientos que debe realizar la persona por cada músculo.

Configuración: Se tiene 4 apartados, donde se especifica el tiempo en segundos que va a realizar cada función; el primero es la cantidad de repeticiones que debe realizar el movimiento indicado por cada músculo, el siguiente es el tiempo-señal el cual es el tiempo el que va adquirir la señal, en la parte derecha está el tiempo-músculo que es el tiempo que demora para el cambio de músculo durante la toma de señales. A continuación, el tiempo-acción que define el tiempo de espera entre las acciones o movimientos a ejecutar, el tiempo-señal que indica el tiempo que las señales serán almacenadas, y por último se tiene Origen/Destino para identificar el directorio donde se guardarán los archivos .csv de los datos.

Además, se tiene los botones para Iniciar, Pausar y Finalizar; por medio de ellos se controla el programa durante todo el proceso.

Monitor de señal: Se pueden observar las señales de los tres sensores (canales) de forma gráfica en tiempo real, mientras se realizan los movimientos descritos. Todas estas configuraciones y el funcionamiento se muestran en la Ilustración 3-8.

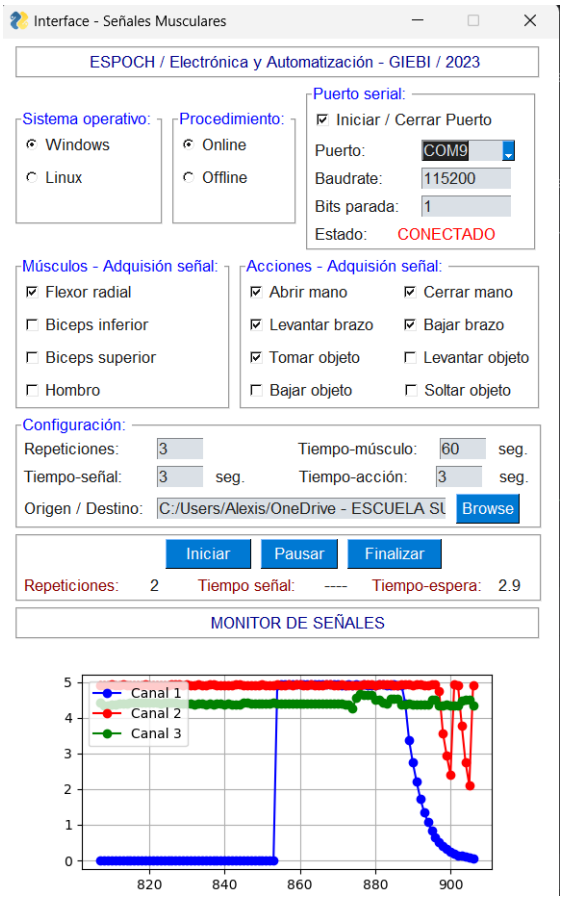


***Ilustración* 3-8** Interfaz gráfica en modo CONECTADO durante la adquisición de señales

Realizado por: Vaca-Paguay, 2024.

3.2.3 Etapa de clasificación y normalización

En la etapa de clasificación y estandarización de señales mioeléctricas, se empieza cargando los archivos .csv obtenidos en la anterior etapa. Empleando la librería Pandas en Python, se procesan y organizan los datos para su posterior análisis. Los archivos se organizan en conjuntos específicos, dividiéndolos según características de acuerdo con cada combinación posible según su tipo de movimiento, sensor, músculo y brazo. Por ejemplo, para el brazo izquierdo, los títulos de las variables podrían seguir un patrón como BI_MT_S1_M1, donde BI indica el brazo izquierdo, MT representa el músculo tensor, S1 el sensor uno y M1 hace referencia al movimiento uno.

En la tabla 3-5 se detalla la nomenclatura utilizada para cada conjunto de señales y sus posibles combinaciones utilizadas en este proyecto.

Tabla 3-5 Descripción de la nomenclatura utilizada para cada conjunto de señales.

Variables	Nomenclatura	Descripción
Brazos	BD	Brazo derecho
	BI	Brazo izquierdo
Sensores	S1	Sensor uno
	S2	Sensor dos
	S3	Sensor tres
Músculos	MT	Músculo tensor
	MBI	Músculo bíceps inferior
	MBS	Músculo bíceps superior
Movimientos	M1	Movimiento abrir mano
	M2	Movimiento cerrar mano
	M3	Movimiento levantar brazo
	M4	Movimiento bajar brazo
	M5	Movimiento tomar objeto
	M6	Movimiento levantar objeto
	M7	Movimiento bajar objeto
	M8	Movimiento soltar objeto

Realizado por: Vaca-Paguay, 2024.

En la Ilustración 3-9 se presenta el diagrama de bloques correspondiente a la fase de clasificación y normalización de datos.

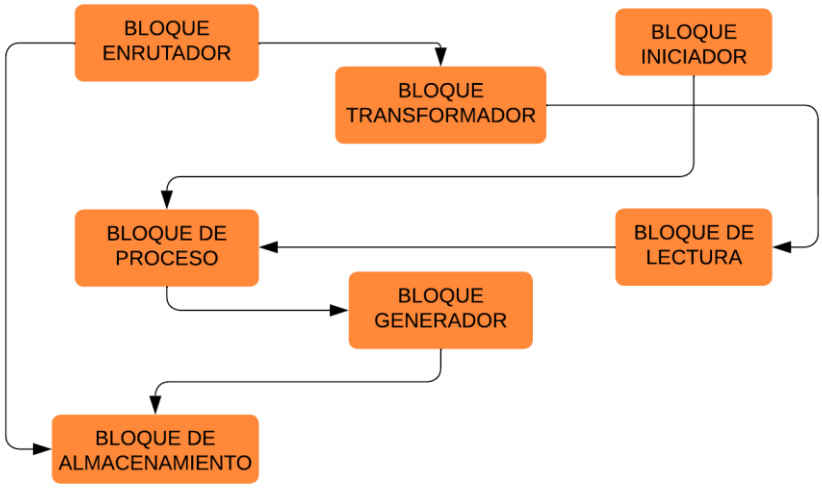


Ilustración 3-9 Diagrama de bloques etapa de clasificación y normalización

Realizado por: Vaca-Paguay, 2024

Bloque enrutador: Representa la ubicación de los archivos CSV que contienen datos de diferentes conjuntos, subdivididos en señales S1, S2, S3, etc. Cada ruta apunta a un conjunto específico de archivos CSV.

Bloque transformador: Este bloque se encarga de modificar las estructuras de datos originales después de la combinación, permitiendo subdividirlos, reorganizarlos o modificarlos para adaptarlos a necesidades específicas. Esto incluye la división de los datos, la normalización, etc.

Bloque iniciador: Se crean listas vacías para almacenar los dataframes divididos en 2 conjuntos obligatoriamente BI (brazo izquierdo) – BD (brazo derecho). Estas listas servirán como contenedores para los dataframes que se generarán más adelante.

Bloque de proceso: Este bloque involucra la manipulación de los datos extraídos de los archivos CSV. Para cada archivo y fila de datos leída, se verifica la existencia de información y se genera un dataframe correspondiente. Estos dataframes se agregan a las listas previamente creadas para almacenar los nuevos conjuntos de datos.

Bloque generador: Se crea un dataframe por cada fila de datos procesados.

Bloque de almacenamiento: Los dataframes resultantes, ya concatenados o procesados, se almacenan en las listas previamente creadas.

3.2.4 Etapa de entrenamiento virtual

Luego de estandarizar los datos de las señales mioeléctricas, el entrenamiento se lleva a cabo utilizando redes neuronales convolucionales (CNN) y redes neuronales secuenciales en el entorno de Google Colab. Esta elección se debe principalmente al acceso abierto a recursos informáticos

de alta capacidad, como GPU y TPU (limitadas), que aceleran significativamente el proceso de entrenamiento del modelo (von Chamier et al., 2021). Las características generales se muestran en la Tabla 3-6.

Tabla 3-6 Especificaciones Google Colab

Especificación	Descripción
Lenguaje de programación	Basado en Python, permitiendo la escritura y ejecución de código Python en celdas interactivas
Ambiente de ejecución	Entorno de cuadernos basado en la nube
Acceso a GPUs/TPUs	Sí, acceso gratuito a GPUs (limitado) y TPUs para aceleración de cómputo
Integración con Drive	Total, integración con Google Drive para almacenamiento y acceso a datos y modelos
Bibliotecas preinstaladas	Viene con bibliotecas populares preinstaladas como TensorFlow, Keras, NumPy, entre otros
Colaboración	Posibilidad de compartir y colaborar en tiempo real con otros usuarios en un mismo cuaderno
Tiempo de ejecución	Limitado (12 horas) antes de reiniciar, con almacenamiento local temporal
Restricciones	Algunas restricciones en la cantidad de recursos disponibles en la versión gratuita

Realizado por: Vaca-Paguay, 2024.

La primera CNN, diseñada para la clasificación categórica de los ocho movimientos posibles, consta de dos capas convolucionales, cada una con 64 filtros y un tamaño de núcleo de 3, activadas por la función *ReLU*. Se añade una capa de eliminación con un porcentaje del 50% para prevenir el sobreajuste, seguida de una capa de MaxPooling con un tamaño de grupo de 2 para reducir la dimensionalidad de los patrones extraídos. Luego, las características resultantes se aplanaron para alimentar una densa capa de 100 neuronas también activadas por una función *ReLU*. Finalmente, se incluye una capa de salida densa con una función de activación *SOFTMAX* para la clasificación categórica de las señales. El modelo se construyó utilizando la función de pérdida *CATEGORICAL_CROSSENTROPY* y el optimizador *Adam*, con métricas de precisión para evaluar el rendimiento general de la red.

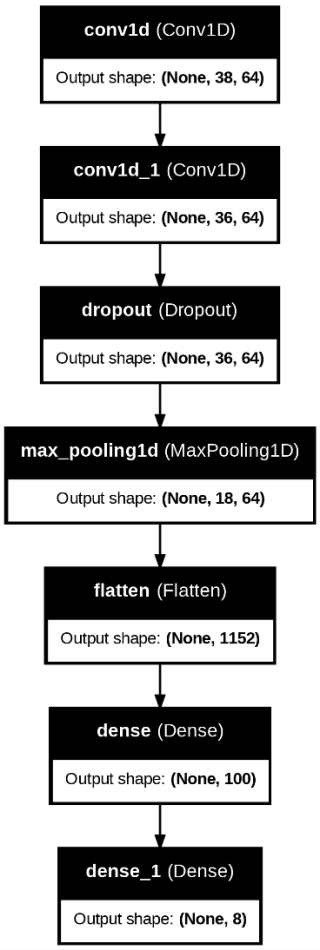


Ilustración 3-10 Arquitectura de la red categórica

Realizado por: Vaca-Paguay, 2024

La segunda CNN, destinada a predecir el comportamiento temporal de las señales mioeléctricas, se estructura con una capa convolucional inicial con 64 filtros y un tamaño del núcleo de 5, seguida de una capa *MAXPOOLING* con un tamaño de 2 para reducir la dimensionalidad. A continuación, se agregan dos capas convolucionales más, cada una con 128 y 256 filtros respectivamente y un tamaño de núcleo de 5, todas activadas por una función ReLU y seguidas por una nueva capa *MAXPOOLING* con un tamaño de grupo de 2. Las características resultantes se aplanaron y se introdujeron en una capa densa. capa con 256 neuronas activadas por ReLU. Para evitar el sobreajuste, se agrega una capa de abandono con una tasa del 50%, seguida de dos capas densas adicionales que contienen 128 y 40 neuronas, respectivamente. El modelo se compiló utilizando la función de pérdida del error cuadrático medio (MSE) y el optimizador Adam para optimizar la función de pérdida.

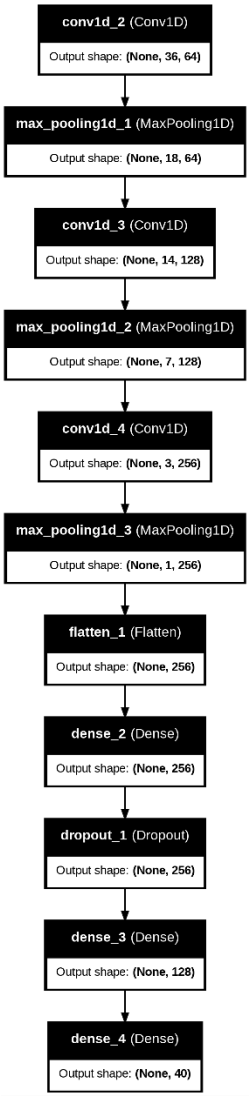


Ilustración 3-11 Arquitectura de la red de predicción temporal

Realizado por: Vaca-Paguay, 2024

En el diagrama de bloques de la Ilustración 3-12 se detalla el proceso de entrenamiento virtual, este entrenamiento virtual es muy importante para el desarrollo y ajuste de los modelos de aprendizaje automático.

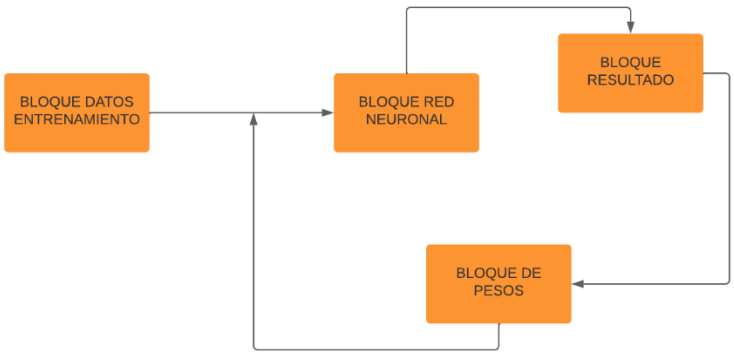


Ilustración 3-5 Diagrama de bloques etapa de entrenamiento virtual

Realizado por: Vaca-Paguay, 2024.

Bloque de datos entrenamiento: Este bloque representa el conjunto de datos utilizado para entrenar la red neuronal. Incluye todas las instancias de entrada y sus correspondientes salidas esperadas (o etiquetas). Es la capa inicial de la red neuronal donde se introducen los datos. Cada nodo en esta capa representa una característica o atributo de los datos de entrada.

Bloque de Red Neuronal (entrenamiento): Este bloque representa la estructura de la red neuronal, incluyendo todas sus capas (como la capa de entrada, capas ocultas y capa de salida), así como la forma en que están conectadas las neuronas en cada capa.

Bloque resultado: Es la capa final de la red neuronal donde se generan las predicciones o resultados luego de que los datos han sido procesados a través de las capas ocultas. Representa la combinación de los resultados de las capas anteriores para obtener la salida final de la red neuronal. En esta etapa se suman o combinan las señales de las neuronas conectadas en las capas anteriores.

Bloque de pesos: Son los parámetros internos de la red neuronal que se ajustan durante el proceso de entrenamiento. Cada conexión entre neuronas tiene un peso que se ajusta para que la red pueda aprender y hacer predicciones precisas.

3.2.5 Etapa evaluación del modelo final de Machine Learning

La evaluación de modelos de redes neuronales convolucionales (CNN) para el análisis de señales mioeléctricas implica un análisis integral de su desempeño en la clasificación y la predicción temporal (Fiallos, 2021).

En la clasificación, se analiza la capacidad del modelo para identificar correctamente diferentes tipos de movimiento a partir de las señales. El análisis se centra en comprender cómo el modelo interpreta y clasifica las características de la señal, así como en identificar posibles errores de

clasificación. Se observa que estos modelos pueden identificar con mayor precisión la mayoría de los movimientos. Aunque esta precisión podría seguir mejorándose.

En la predicción temporal, por el contrario, se evalúa la capacidad del modelo para predecir el comportamiento futuro de la señal EMG. El análisis implica examinar cómo el modelo captura e infiere las tendencias de las señales, así como identificar posibles diferencias entre las predicciones del modelo y los datos reales. Se observa que, en la mayoría de los casos, el modelo es capaz de predecir con precisión el comportamiento temporal de la señal, lo que indica buenas capacidades de generalización. (Del Brío, 2020). En el diagrama de bloques de la Ilustración 3-13 se representa la etapa de evaluación del modelo resultante, se analiza el rendimiento del modelo de aprendizaje automático entrenado, comparando sus predicciones con los datos de prueba previamente separados.

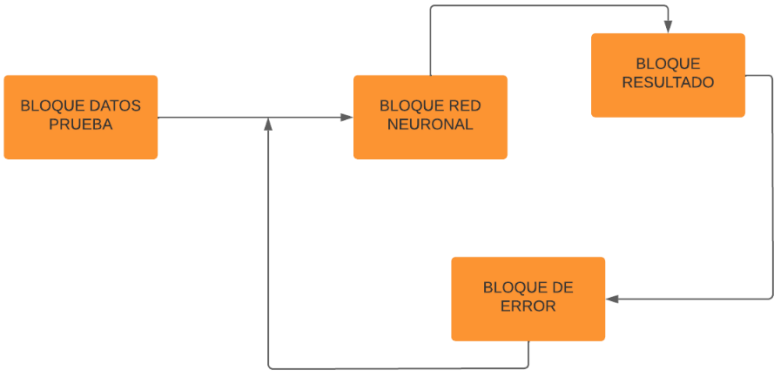


Ilustración 3-6 Diagrama de bloques etapa evaluación del modelo resultante

Realizado por: Vaca-Paguay, 2024.

Bloque de datos pruebas: Para evaluar el modelo, se requiere un conjunto de datos separado no utilizado durante el entrenamiento, llamado conjunto de prueba, para medir su rendimiento en situaciones no vistas previamente (Schlotthauer, 2022). En este caso, se empleará la división aleatoria, que garantiza conjuntos representativos y una evaluación imparcial. Para ello, los datos se mezclan aleatoriamente y se asigna un porcentaje al conjunto de prueba, utilizando el resto para el entrenamiento del modelo.

Bloque Red Neuronal (predicciones): Los datos de prueba se ingresan al modelo entrenado en lotes. Este proceso se repite para todas las muestras del conjunto de prueba. El modelo procesa los datos de prueba y genera predicciones para cada instancia de datos. Se compara las predicciones del modelo con las respuestas reales conocidas en los datos de prueba utilizando

métricas específicas (Schlotthauer, 2022). Estas métricas pueden incluir el error cuadrático medio, coeficiente de determinación entre otros.

Bloque de resultados: Con base en las métricas obtenidas, se puede la precisión del modelo, lo que ofrece información sobre la capacidad del modelo para adaptarse a datos no vistos durante el entreno.

Bloque de error: Si se detecta algún error importante, se podrá ajustar hiperparámetros, modificar la arquitectura de la red neuronal o aplicar técnicas de regularización para mejorar el rendimiento del modelo.

3.3 Desarrollo de hardware

3.3.1 Diseño de carcasas para sensores

Fusion 360 originalmente es un software CAD utilizado básicamente para diseñar piezas mecánicas y físicas. (Vel Tech Rangarajan Sagunthala & Mahato, 2021). Permite crear modelos paramétricos, donde los cambios en las dimensiones y características del modelo se pueden realizar fácilmente mediante la modificación de parámetros predefinidos, ofrece herramientas intuitivas de extrusión, cortes para aberturas y otros detalles que facilitan la creación de las carcasas para los tres sensores MyoWare que se ocupan para la toma de datos como se puede observar en la Ilustración 3-14.

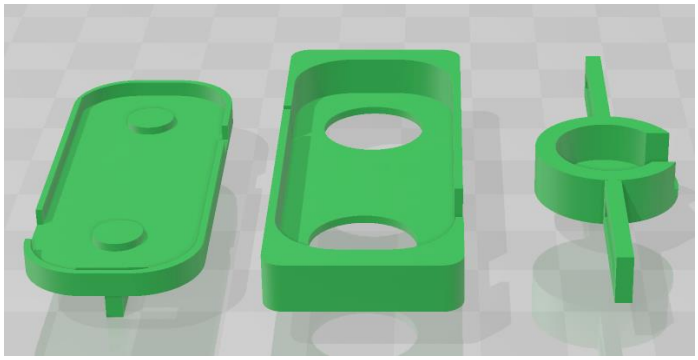


Ilustración 3-7 Diseño carcasas sensores en Fusion 360

Realizado por: Vaca-Paguay, 2024.

3.3.2 Diseño de PCB

KiCad, en combinación con hardware de código abierto, tiene el potencial de desempeñar un papel significativo en la innovación tecnológica, (Quintero Ávalo, 2023) ya que ha sido fundamental en la creación del esquemático y la PCB para el sistema de adquisición de señales. Tabla 3-6. Facilita la representación visual estructurada de los componentes, como la tarjeta de desarrollo ESP-32S y los sensores MyoWare, y simplifica la disposición tridimensional y la traza de conexiones en la placa. Con KiCad, se ha logrado una integración eficiente de todos los

elementos electrónicos esenciales para el sistema del proyecto como se muestra en la Ilustración 3-15.

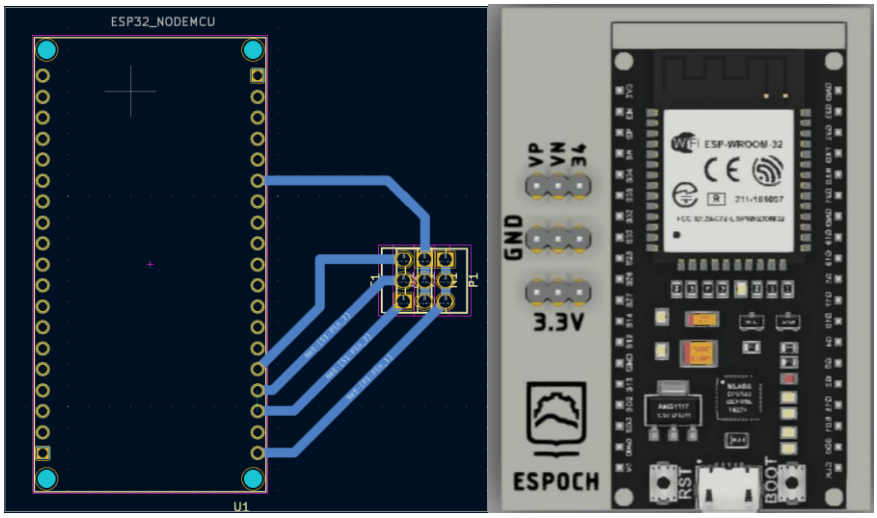


Ilustración 3-8 Diseño PCB placa de adquisición de datos

Realizado por: Vaca-Paguay, 2024.

Tabla 3-7 Características destacadas de KitCad

Características	KiCad
Licencia	Código abierto y gratuito
Plataformas Soportadas	Windows, Linux, macOS
Herramientas Integradas	Esquemático, PCB, Footprint Editor, etc.
Bibliotecas Comunitarias	Amplias bibliotecas de componentes
Desarrollo Activo	Desarrollo activo y actualizaciones frecuentes

Realizado por: Vaca-Paguay, 2024.

3.3.3 Diseño de estructura

Para el diseño de la estructura donde se encuentran todos los componentes electrónicos, que se observa en la Ilustración 3-16, se utiliza la herramienta de SolidWorks, la cual cuenta con una interfaz que facilita la creación y edición de modelos 3D basados en relaciones y dimensiones. Se pueden ajustar fácilmente los parámetros para modificar la forma y tamaño de la caja según sea necesario. Este diseño está impreso con un material de bajo costo y resistente como es el PLA. (Fuentes, 2021)

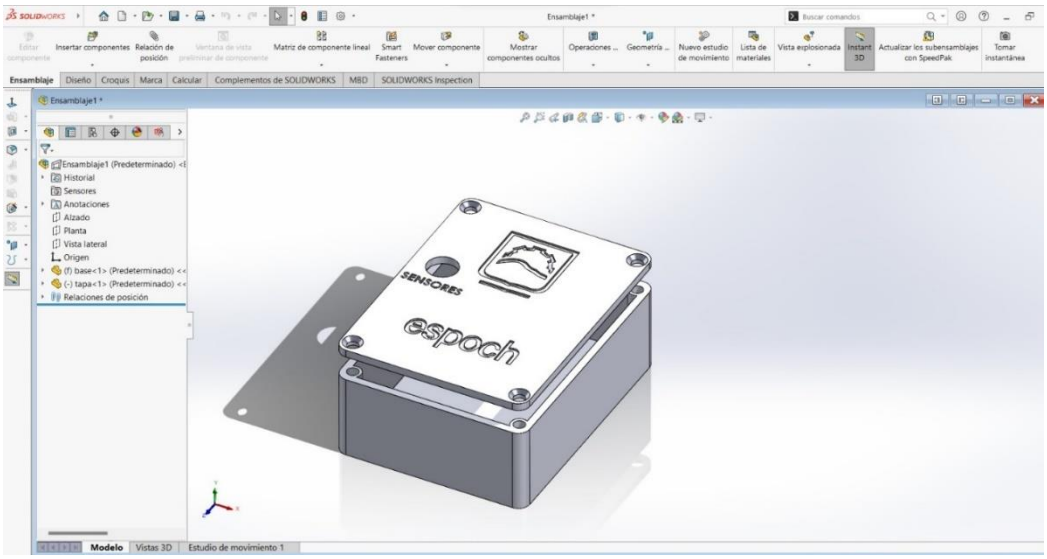


Ilustración 3-9 Diseño en SolidWorks

Realizado por: Vaca-Paguay, 2024.

3.4 Desarrollo de Software

3.4.1 Flujograma adquisición de señales

Tabla 3-8 Funciones relevantes de control

Código	Descripción
Inicialización de variables Ch1	Declara constantes para los pines GPIO que representan las señales de los tres canales.
Inicio comunicación serial	Configuración inicial que establece los pines y la comunicación serial.
Lectura ADC	Bucle principal que lee, convierte y envía datos a través del puerto serial a una frecuencia dada.
Concatenación de datos V1, V2, V3	Construye una cadena de datos con los voltajes de los canales.
Serial.print(data);	Envía la cadena de datos por el puerto serial.

Realizado por: Vaca-Paguay, 2024.

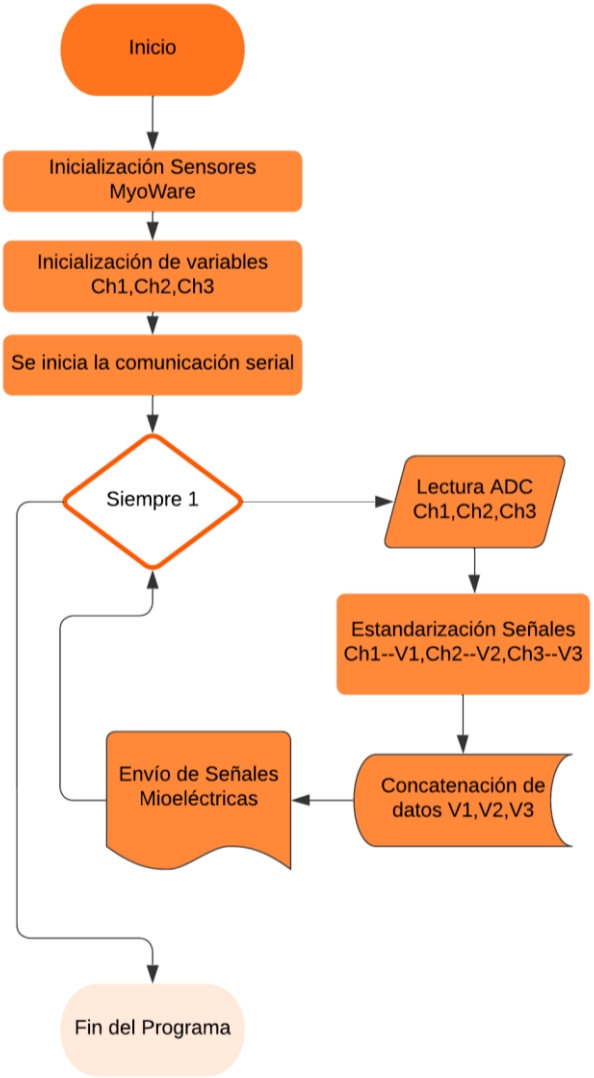


Ilustración 3-10 Flujograma sistema de adquisición de señales

Realizado por: Vaca-Paguay, 2024.

El programa inicia en Arduino IDE configurando los pines Ch1, Ch2 y Ch3 como entradas analógicas. Cualquier comunicación serial existente se detiene e inicializa el puerto serial existente. Ingresa en el bucle principal que ejecuta continuamente las siguientes acciones: lee las señales analógicas de Ch1, Ch2 y Ch3, convierte estos valores a voltajes (V1, V2 y V3), construye una cadena de datos en formato CSV concatenando estos valores, imprime la cadena en el puerto serial y espera 100 milisegundos antes de la próxima lectura y envío de datos. Este proceso se repite de forma continua hasta que el programa se detenga.

3.4.2 Flujograma adquisición, clasificación y visualización de señales

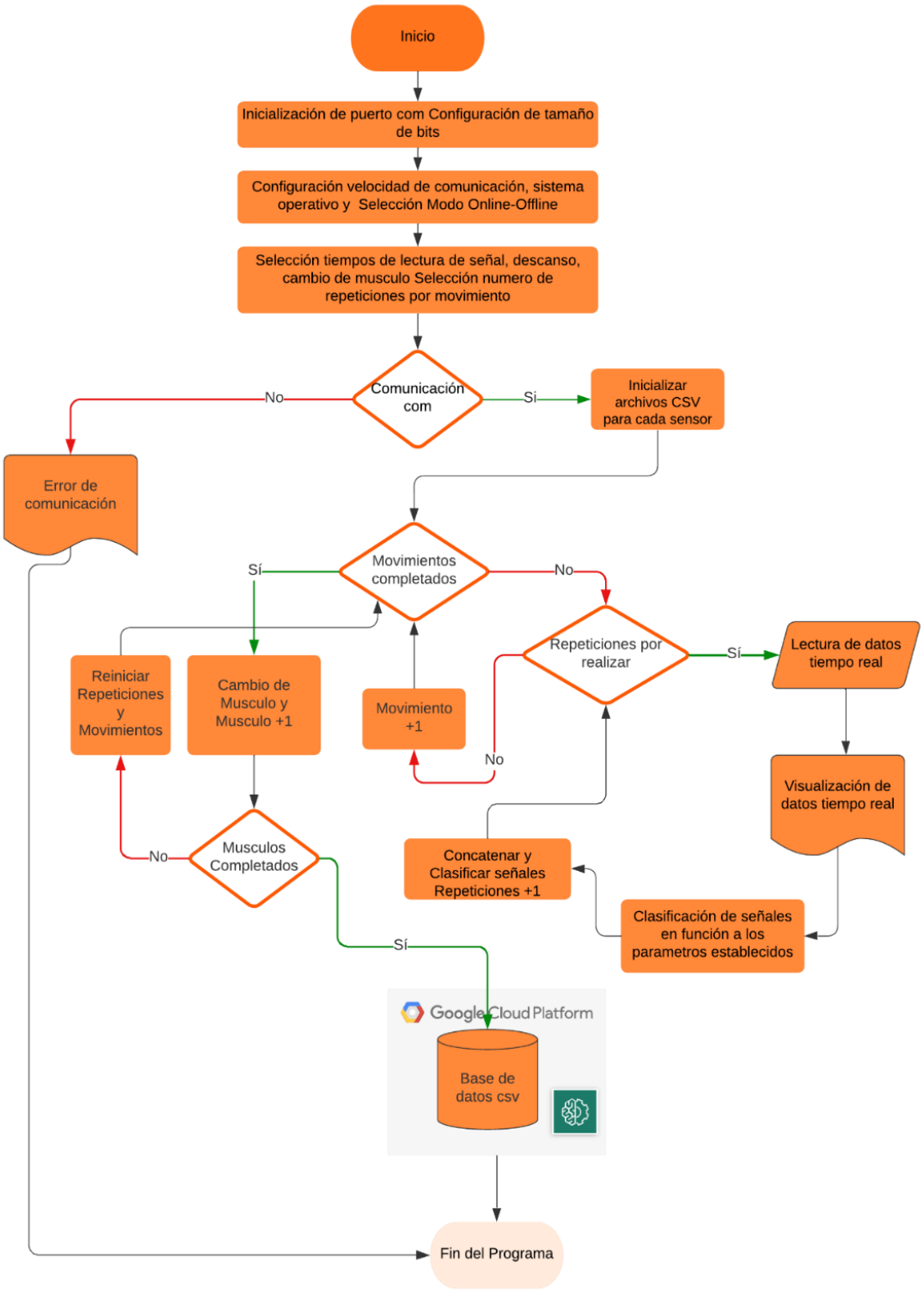


Ilustración 3-11 Flujograma de la interfaz de adquisición, clasificación y visualización de señales.

Realizado por: Vaca-Paguay, 2024.

El flujo de la aplicación comienza con la configuración inicial de parámetros, donde el usuario especifica detalles como los sensores, movimientos, músculos y la cantidad deseada de repeticiones para cada movimiento. Luego, se establece la comunicación UART con la tarjeta de desarrollo para recibir datos mioeléctricos en tiempo real. Una vez completada esta conexión, se inicia la generación de archivos CSV, creando archivos dedicados para cada sensor, músculo y movimiento.

Después, se entra en el ciclo principal diseñado para abordar cada movimiento específico. Dentro de este ciclo, se ejecuta otro bucle para gestionar las repeticiones asociadas a ese movimiento en particular. En cada repetición, la aplicación adquiere datos mioeléctricos en tiempo real, los combina y los guarda en el archivo CSV correspondiente, teniendo en cuenta tanto el movimiento como la repetición actual (Quintero Ávalo, 2023).

Este proceso continúa hasta alcanzar el número deseado de repeticiones para el movimiento específico, después la aplicación sigue al siguiente músculo, repitiendo el mismo procedimiento para todos los movimientos, por último, se reinicia el contador de repeticiones continuando el ciclo. Este ciclo de operaciones persiste hasta que se completan todas las repeticiones para todos los músculos y movimientos configurados, proporcionando una aplicación integral y eficiente para la adquisición y almacenamiento de datos mioeléctricos. (Del Carmen & Ramos, 2021)

Tabla 3-9 Flujo operativo simplificado

Descripción
1. Configuración Inicial: Definir parámetros clave.
2. Comunicación UART: Conectar y recibir datos.
3. Generación de Archivos CSV: Crear archivos.
4. Bucle Principal: Procesar movimientos y repeticiones.
5. Transición a Siguiente Músculo: Cambiar de enfoque.
6. Ciclo Persistente: Repetir hasta completar todas las repeticiones para músculos y movimientos.

Realizado por: Vaca-Paguay, 2024.

3.4.3 Flujograma del sistema de lectura, normalización y entrenamiento virtual

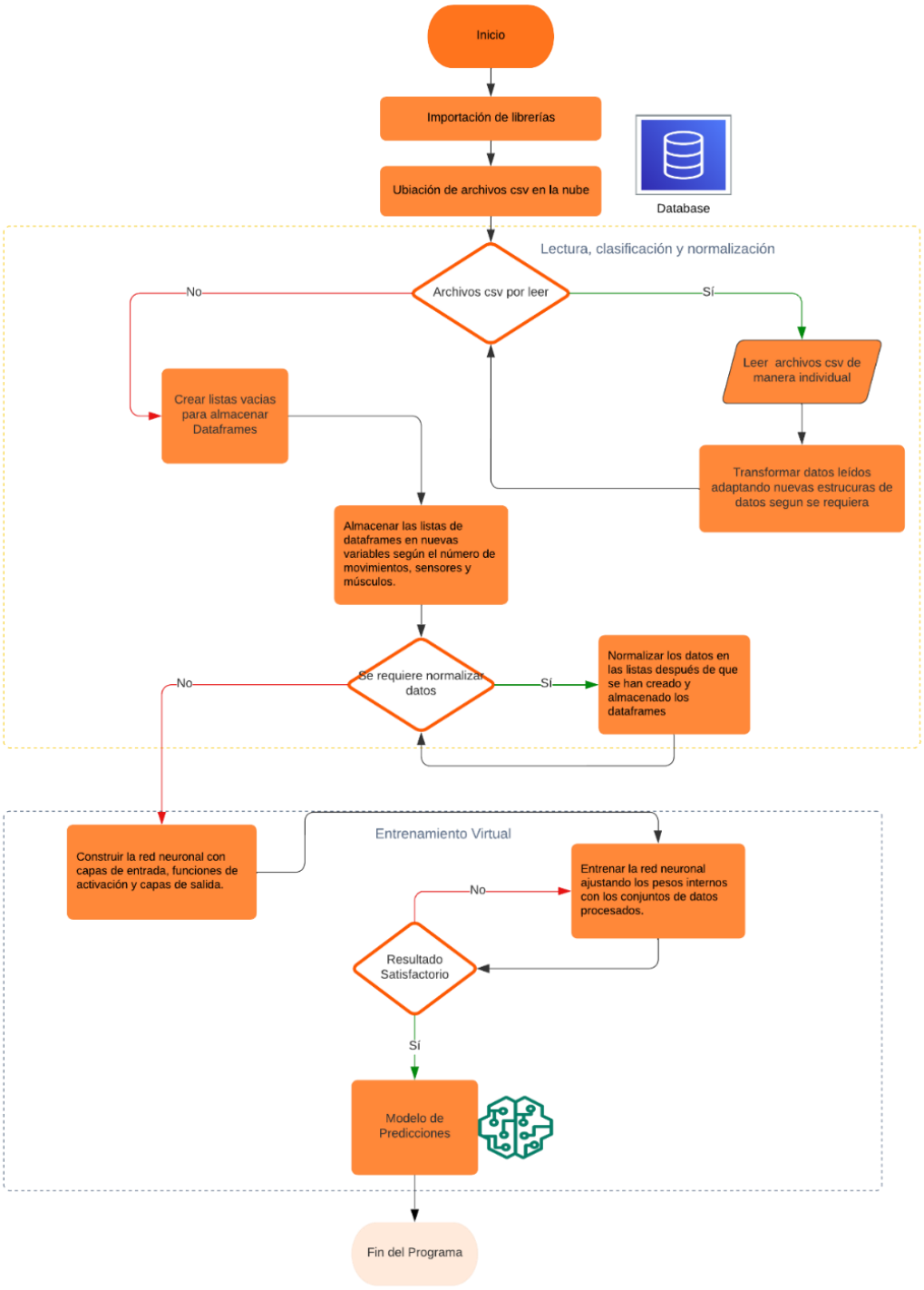


Ilustración 3-12 Flujograma del sistema de lectura, normalización y entrenamiento virtual

Realizado por: Vaca-Paguay, 2024.

En la fase de procesamiento y entrenamiento de datos, los archivos CSV se localizan en la nube, se leen y adaptan según requisitos específicos. Los datos procesados se organizan en listas de dataframes para brazo izquierdo y derecho, almacenadas en variables según movimientos,

sensores y músculos. Se decide si es necesario normalizar los datos antes de construir y entrenar la red neuronal (Del Carmen & Ramos, 2021). La evaluación del modelo con un conjunto de datos de prueba permite ajustes adicionales si es necesario mejorar el rendimiento. Este flujograma da una visión general de la última etapa del proyecto.

3.5 Sistemas Integrados Adicionales

3.5.1 Sistemas de comunicaciones adecuados

La implementación efectiva de un sistema integral de adquisición, clasificación, visualización y exportación de datos se basa en la sincronización de tres protocolos de transferencia clave. Estos protocolos son fundamentales para una comunicación eficiente entre los componentes del sistema, como se detalla en la Ilustración 3-19. Su combinación garantiza una gestión eficiente de la información, clave para su aplicación final.

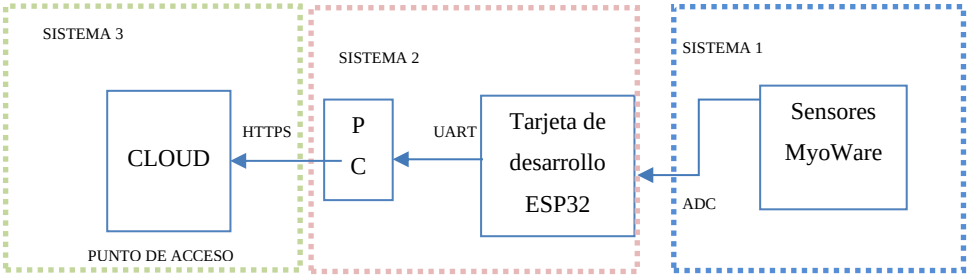


Ilustración 3-13 Diagrama de comunicación del sistema

Realizado por: Vaca-Paguay, 2024.

En el primer sistema, se realiza la adquisición de datos analógicos procedentes de sensores mediante el uso del convertidor analógico a digital (ADC) propia de la tarjeta de desarrollo ESP32, con una capacidad de hasta 12 bits de resolución, facilita la captura detallada de información, sentando así la base y origen de información para su posterior procesamiento. En el segundo sistema, la tarjeta de desarrollo establece una conexión UART para la comunicación con la PC. A través de esta conexión serial, la información procesada se transfiere desde la tarjeta de desarrollo al buffer serial, listo para ser enviado a la interfaz en la PC. El uso de UART garantiza una transferencia eficiente y confiable de datos, facilitando la integración fluida entre ambos entornos. Por último, en el tercer sistema, los datos en formato CSV se envían de la PC a la nube mediante HTTPS. Este enfoque asegura una transferencia segura y confiable, fomentando la disponibilidad remota y accesibilidad de los conjuntos de datos archivados. Esta metodología respalda la interoperabilidad y eficiencia en la gestión integral de datos del sistema (González, 2023).

3.6 Esquema de conexiones

3.6.1 Sistema de adquisición de datos

En la ilustración 3-19, se muestra el esquemático que detalla la conexión entre la tarjeta de desarrollo ESP32S y los sensores MyoWare. En esta configuración, la ESP32S funge como el controlador principal, recibiendo datos desde los sensores MyoWare a través de sus pines GPIO asignados.

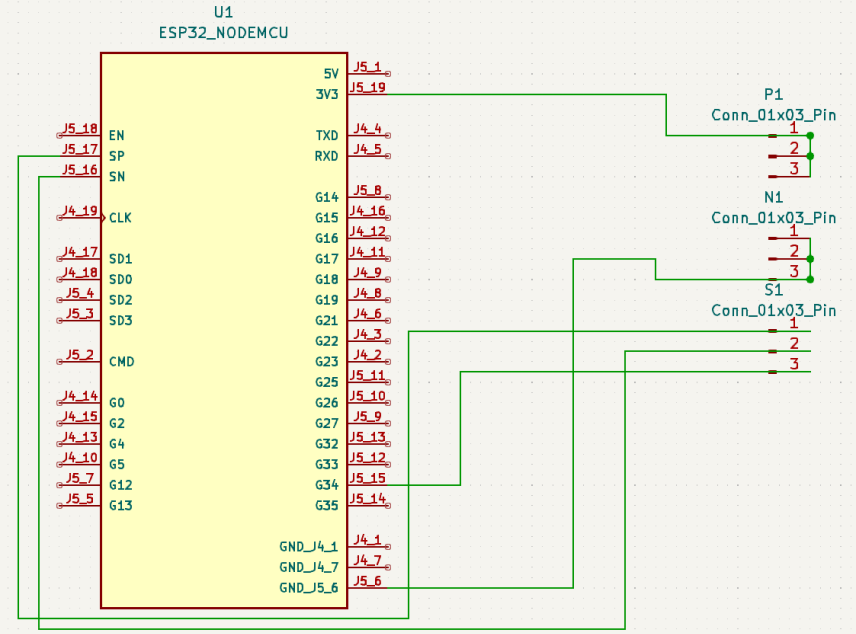


Ilustración 3-14 Diagrama de conexiones del sistema de adquisición de datos

Realizado por: Vaca-Paguay, 2024.

3.3V (Pin de Alimentación): conecta el pin de 3.3V de la tarjeta de desarrollo ESP32S al pin de alimentación de los sensores MyoWare para suministrarles energía.

GND (Tierra): conecta el pin GND de la tarjeta de desarrollo ESP32S al pin GND de los sensores MyoWare para cerrar el circuito de alimentación y establecer una referencia común de tierra.

GPIO 39 - Lectura Sensor 1: conecta el pin GPIO 34 de la tarjeta de desarrollo ESP32S al pin de lectura del primer sensor MyoWare para capturar los datos analógicos generados por ese sensor.

GPIO 36 - Lectura Sensor 2: conecta el pin GPIO 39 de la tarjeta de desarrollo ESP32S al pin de lectura del segundo sensor MyoWare para adquirir los datos analógicos correspondientes a ese sensor.

GPIO 34 - Lectura Sensor 3: conecta el pin GPIO 36 de la tarjeta de desarrollo ESP32S al pin de lectura del tercer sensor MyoWare para obtener los datos analógicos del tercer sensor.

3.7 Implementación del sistema

En la Ilustración 3-21, se presenta la implementación integral de todos los sistemas y componentes esenciales para llevar a cabo la adquisición, procesamiento, visualización y almacenamiento de datos de señales mioeléctricas, según se detalla en este capítulo.

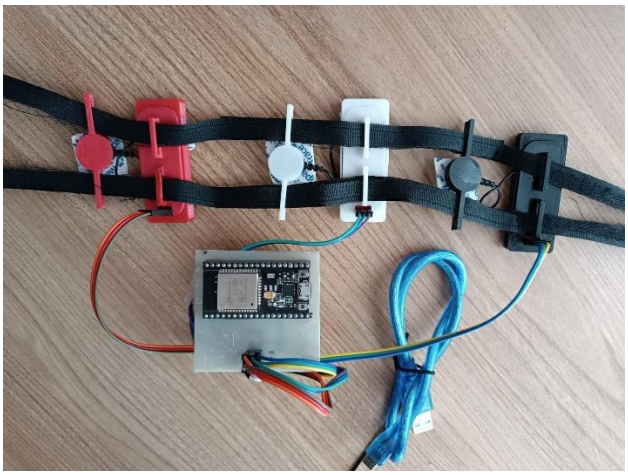


Ilustración 3-21 Elementos hardware integrados para la implementación

Realizado por: Vaca-Paguay, 2024.

CAPÍTULO IV

4. ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

En este apartado se presentan las pruebas realizadas para evaluar el funcionamiento del sistema, con las cuales se permite demostrar la precisión y el rendimiento

4.1 Validación de sensores mioeléctricos

4.1.1 Validación de los sensores para garantizar su precisión y fiabilidad.

Para la validación de los sensores mioeléctricos MyoWare, se ha realizado una prueba con un multímetro para comparar el voltaje, la corriente y la ganancia entre la hoja de datos del sensor MyoWare Anexo D y la práctica real del sensor, como se puede visualizar en la siguiente Tabla 4-1.

Tabla 4-1 Validación de sensores

Parámetros	Valores reales del sensor			Valores de la hoja de datos del sensor		
	Ganancia (kΩ)	Voltaje (V)	Corriente (mA)	Ganancia (kΩ)	Voltaje (V)	Corriente (mA)
Min.	0.004	3.289	------	0.01	+2.9	------
TYP	50	3.355	9.32	50	+3.3 o +5	9
Máx.	99.5	3.160	13.15	100	+5.7	14

Realizado por: Vaca-Paguay, 2024.

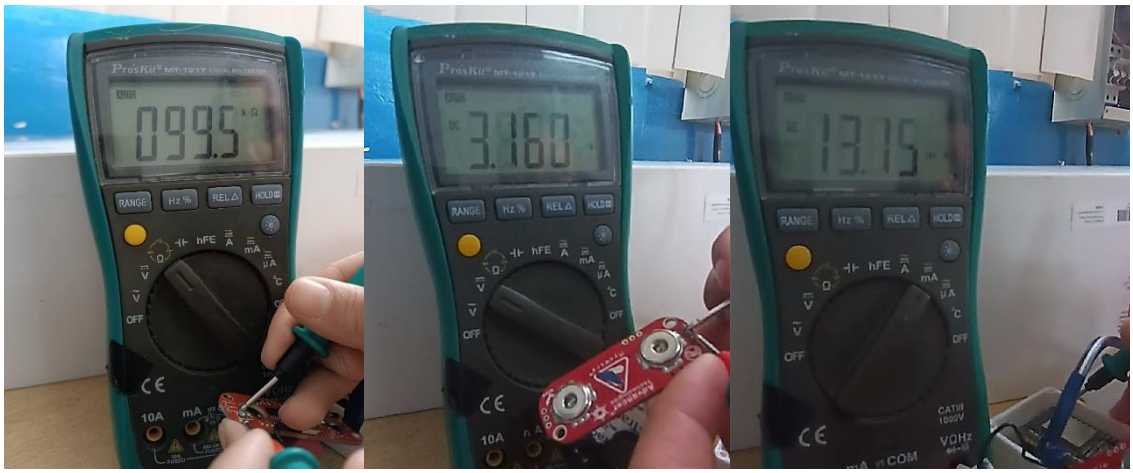


Ilustración 4-1 Comprobación de parámetros del sensor MyoWare

Realizado por: Vaca-Paguay, 2024.

Una vez validado el sensor, se identifica que existe mínimos errores donde por ejemplo la ganancia máxima es menor con 0.5kΩ, por otro lado, las corrientes y sus voltajes van en el rango establecido de acuerdo con la hoja de datos.

4.1.2 Prueba de variación de ganancia para verificar el funcionamiento

Se ha realizado una prueba con un solo sensor como se muestra en la Ilustración 4-2. Al ejecutar la acción de abrir y cerrar mano se ha comparado con dos valores de ganancia de 25kΩ y de 75kΩ. En la Tabla 5-2, se muestra la variación de la señal con alta y baja resistencia en el sensor.

Ilustración 4-2 Prueba de variación de ganancia

Realizado por: Vaca-Paguay, 2024.

Tabla 4-2 Comparación de variación de ganancia

Movimiento	Ganancia 25kΩ	Ganancia 75kΩ
Abrir mano	3,43V	0,72V
	3,45V	0,71V
	3,44V	0,7V
	3,46V	0,71V
	3,36V	0,65V
	3,37V	0,62V
	3,4V	0,78V
	3,39V	0,59V
	3,27V	0,59V
	3,19V	0,64V
Cerrar mano	4,42V	1,61V
	4,46V	1,19V
	4,56V	1,21V
	4,62V	1,28V
	4,78V	2,24V
	4,75V	1,9V
	4,81V	1,15V
	4,85V	1,02V
	4,89V	1,14V
	4,93V	1.19V

Realizado por: Vaca-Paguay, 2024.

Como se observa en la Tabla 4-2, se disponen de 20 datos, 10 para el movimiento de abrir mano y 10 para el de cerrarla, se observa que existen diferentes rangos y valores de voltajes medidos en la salida, por lo cual se ha realizado una prueba estadística t Student que compara las medias de una variable continua clasificada según otras dos categorías de una variable nominal (Chen, Wang, & Gorban, 2019) para evaluar su diferencia con la definición de las siguientes hipótesis:

H0: No hay diferencia significativa entre las dos ganancias para los dos movimientos.

H1: Hay una diferencia significativa entre las dos ganancias para los dos movimientos.

Después de tener las hipótesis, la prueba t Student para muestras independientes se calcula la prueba t utilizando la Ecuación 3.

$$t = \frac{x_1 - x_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \quad (3)$$

Donde, x_1 y x_2 son las medias de los dos grupos, s_1 y s_2 son las desviaciones estándar de los dos grupos y n_1 y n_2 son los tamaños de muestra de los dos grupos.

Tabla 4-3 Resultados de la prueba t-Student

Ganancias	Movimiento	Media	Desviación estándar
25kΩ	Abrir mano	3,399	0,080
	Cerrar mano	4,742	0,209
75kΩ	Abrir mano	0,674	0,058
	Cerrar mano	1,453	0,383

Realizado por: Vaca-Paguay, 2024.

En la Tabla 4-3 se observa los resultados de las medias y desviación estándar de las dos ganancias para los dos movimientos, además se tiene en cuenta que los tamaños de muestra de los dos grupos es 10. Con estos datos el valor t es de aproximadamente -10,64. El grado de libertad es de 18 con un grado de significancia de 0,05. Se obtiene el valor crítico de t para dos colas mediante una tabla de distribución que es de 2,101. El valor absoluto de t es mayor que el valor crítico, por lo tanto, se rechaza la hipótesis nula (H0) y se concluye que efectivamente hay una diferencia significativa entre las dos ganancias para los dos movimientos.

4.1.3 Evaluación de la respuesta del sensor ante diferentes niveles de actividad muscular o movimientos específicos.

Para la evaluación de la respuesta del sensor se realiza una recolección de 10 datos por movimiento. La tabla completa con los resultados se encuentra en el Anexo F, esta prueba consiste en comparar si existe correlación (Fiallos, 2021) , entre los tres sensores mioeléctricos que rodean la zona del bíceps inferior como se muestra en la Ilustración 4-3 y realizar 8 diferentes movimientos específicos los cuales fueron abrir y cerrar mano, levantar y bajar brazo, agarrar objeto, levantar y bajar objeto y soltar objeto. Los resultados generales se muestran en la Tabla 4-4.

Ilustración 4-3 Prueba de diferentes movimientos

Realizado por: Vaca-Paguay, 2024.

Tabla 4-4 Comprobación de respuesta ante diferentes movimientos específicos

Movimiento	Sensor 1	Sensor 2	Sensor 3
Abrir mano	1,72V	0,95V	2,6V
	1,73V	0,92V	3,55V
	1,79V	1,0V	3,6V
	1,85V	1,02V	3,63V
	2,3V	1,29V	3,87V

Cerrar mano	4,53V	2,49V	4,67V
	3,67V	2,35V	4,58V
	3,41V	2,12V	4,47V
	3,09V	1,94V	4,32V
	2,72V	1,77V	4,21V
Levantar brazo	3,41V	1,4V	3,61V
	2,94V	1,06V	3,43V
	2,17V	1,01V	2,84V
	1,48V	0,98V	2,67V
	1,05V	0,94V	2,08V
Bajar brazo	1,31V	0,79V	1,73V
	1,32V	0,74V	1,68V
	1,36V	0,71V	1,62V
	1,23V	0,69V	1,57V
	1,03V	0,67V	1,55V
Agarrar objeto	2,76V	1,06V	3,53V
	2,43V	1,01V	3,28V
	2,0V	0,94V	3,12V
	1,48V	0,76V	2,89V
	1,05V	0,64V	2,62V
Levantar objeto	4,24V	2,32V	4,43V
	3,59V	2,01V	4,38V
	3,38V	1,82V	4,31V
	2,17V	1,78V	4,29V
	1,61V	1,64V	3,63V
Bajar objeto	1,66V	1,17V	2,24V
	1,43V	1,04V	2,21V
	1,29V	0,77V	2,14V
	1,24V	0,68V	1,94V
	1,19V	0,56V	1,8V
Soltar objeto	0,96V	0,53V	1,72V
	0,86V	0,49V	1,68V
	0,84V	0,46V	1,63V
	0,73V	0,44V	1,57V
	0,65V	0,39V	1,42V

Realizado por: Vaca-Paguay, 2024.

Los datos de la tabla anterior se utilizaron para determinar los rangos de cada sensor y cada movimiento. Los resultados se presentan en la Tabla 4-5.

Tabla 4-5 Rangos de los diferentes movimientos

Movimiento	Sensor 1	Sensor 2	Sensor 3
Abrir mano	[1,72 – 2,51]	[0,95 – 1,63]	[2,6 – 4,22]
Cerrar mano	[2,29 – 4,53]	[1,47– 2,49]	[3,74 – 4,67]
Levantar brazo	[1,05 – 3,41]	[0,9 – 1,4]	[2,28 – 3,61]
Bajar brazo	[0,76 – 1,31]	[0,37– 0,79]	[1,26 –1,73]
Agarrar objeto	[1,05 – 2,76]	[0,20 – 1,06]	[2,34 – 3,53]
Levantar objeto	[1,45 – 4,24]	[1,19 – 2,32]	[3,29 – 4,43]
Bajar objeto	[0,74 – 1,66]	[0,12 – 1,17]	[1,39 – 2,24]
Soltar objeto	[0,29 – 0,96]	[0,10 – 0,53]	[0,98 – 1,72]

Realizado por: Vaca-Paguay, 2024.

Se concluye que los rangos son distintos para cada movimiento lo que indica que varían dependiendo de la acción realizada, también podemos notar que los valores difieren entre los tres sensores, lo que se interpreta que cada sensor capta la señal debido a su ubicación y sensibilidad. Además, estos resultados nos ayudan a la identificación y clasificación de movimientos basados en las señales captadas por cada sensor a la hora de realizar nuestro algoritmo de Machine Learning.

4.2 Validación del software de adquisición de señales

4.2.1 Prueba de integridad de datos

En esta prueba se verifica la consistencia y completitud de datos almacenados en los archivos. (Caro Zapata, 2022)

Para llevar a cabo la evaluación, se analizan las listas de datos cargadas para el brazo izquierdo y derecho, respectivamente. Cada una de estas listas contiene datos de la combinación de tres tipos de músculos: el Músculo Tensor, el Músculo Bíceps Inferior (MBI) y el Músculo Bíceps Superior (MBS), así como de sus tres sensores ubicados proporcionalmente en cada zona. Además, la combinación de ocho movimientos diferentes genera 72 variables para cada brazo, teniendo 144 variables en total. Se emplea el método. 'isnull **()'** para detectar valores faltantes en los dataframes. Si se identifica algún valor faltante, se muestra la distribución de estos mediante un gráfico de barras. La ausencia de datos faltantes indica la integridad de los datos, mientras que la presencia de estos requiere una investigación para corregir posibles problemas en el proceso de adquisición o manipulación de datos. (Caro Zapata, 2022).

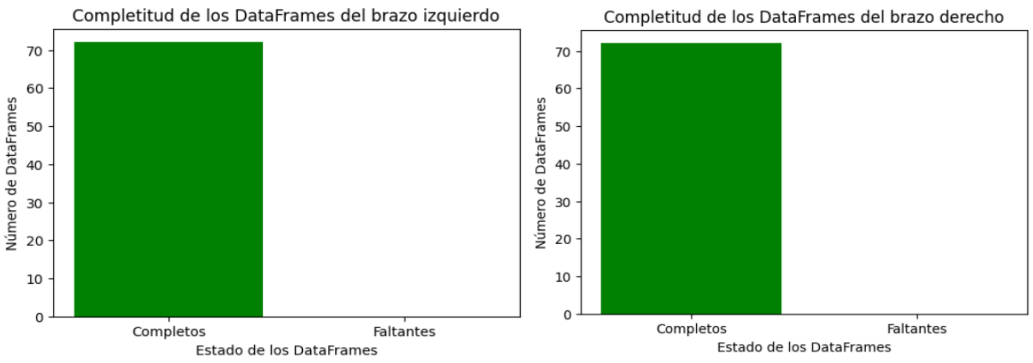


Ilustración 4-4 Diagrama de barras completitud para dataframes

Realizado por: Vaca-Paguay, 2024.

Después de realizar la prueba de integridad de datos en las listas de brazos izquierdo y derecho, se determina que ambas listas estaban completas, es decir, no se detectaron datos faltantes en ninguna de las variables analizadas. Esto indica que la adquisición y almacenamiento de datos se realizó correctamente, asegurando que los datos son consistentes y confiables.

4.2.2 Prueba de coherencia temporal

La coherencia temporal en las variables de las señales mioeléctricas es fundamental para asegurar que los datos recopilados reflejen con precisión la actividad eléctrica de los músculos en un intervalo de tiempo específico.(Caro Zapata, 2022). En esta prueba, se verifica si todas las filas de los Dataframes tienen la misma cantidad de columnas, lo que indica que cada fila representa una cantidad constante de datos tomados en un período de tiempo determinado. En el caso de la aplicación de adquisición de datos que estamos considerando, se limita el tiempo de toma de señal a 3 segundos. Además, en el programa de la tarjeta de desarrollo utilizado para la lectura de los sensores, se establece un intervalo de 75 milisegundos entre lecturas de sensor y envío de datos, garantizando así una frecuencia de muestreo adecuada para los sensores mioeléctricos. La Ecuación 4 describe el cálculo de número de muestras de los sensores con los intervalos definidos, se espera realizar aproximadamente 40 lecturas de sensor en un período de 3 segundos.

$$Número\ de\ muestras = \frac{Tiempo\ de\ adquisicón\ de\ señal}{Tiempo\ de\ espera\ entre\ envio\ de\ datos}$$

$$Número\ de\ muestras = \frac{3000\ ms}{75\ ms} = 40 \quad (4)$$

Ilustración 4-5 Gráfica resultante de la prueba de coherencia temporal

Realizado por: Vaca-Paguay, 2024.

Tabla 4-6 Resumen análisis de coherencia temporal

Brazo	Dataframes	Datos Esperados (cada dataframe)	Inconsistencia
Derecho	72	40	Falso
Izquierda	72	40	Falso

Realizado por: Vaca-Paguay, 2024.

En la Tabla 4-6 se muestra los resultados de la prueba, no se encontraron inconsistencias en ninguna de las muestras, lo que sugiere una coherencia temporal adecuada en los datos de ambos brazos.

4.3 Análisis de datos

4.3.1 Análisis de correlación entre sensores para el mismo musculo y movimiento

La correlación se utiliza para analizar la relación entre los sensores ubicados en la misma zona muscular y utilizados para el mismo tipo de movimiento en ambos brazos. Su objetivo es determinar si existe una correlación significativa entre las señales captadas por cada combinación posible de sensores.

El coeficiente de correlación de Pearson (Fiallos, 2021) se calcula para cada combinación de sensores y movimiento. Este coeficiente varía entre -1 y 1, donde 1 indica una correlación positiva perfecta, -1 una correlación negativa perfecta y 0 ausencia de correlación lineal.

$$rXY = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \quad (5)$$

Donde, rXY es el coeficiente de Pearson, X_i y Y_i son valores individuales de las variables y $\bar{X}$ y $\bar{y}$ son las medias de las variables

Los resultados completos de las correlaciones obtenidas realizadas con la Ecuación 5 para cada movimiento analizado se presentan en el Anexo G.

Además, para facilitar la comprensión de los resultados, se presentarán los coeficientes de correlación de las cinco primeras combinaciones como se muestra en la Tabla 4-7. Estos resultados preliminares se utilizarán para explicar la tendencia observada en la relación entre las señales de los sensores en cada combinación para ambos brazos.

Tabla 4-7 Análisis de correlación entre sensores (brazo derecho)

N°	Combinaciones	Coeficiente de correlación
1	'BD_MT_S1_M1', 'BD_MT_S2_M1', 'BD_MT_S3_M1'	0.675
2	'BD_MT_S1_M2', 'BD_MT_S2_M2', 'BD_MT_S3_M2'	0.739
3	'BD_MT_S1_M3', 'BD_MT_S2_M3', 'BD_MT_S3_M3'	0.649
4	'BD_MT_S1_M4', 'BD_MT_S2_M4', 'BD_MT_S3_M4'	0.614
5	'BD_MT_S1_M5', 'BD_MT_S2_M5', 'BD_MT_S3_M5'	0.721

Realizado por: Vaca-Paguay, 2024.

En la Ilustración 4-6 se observan las correlaciones de las cinco primeras combinaciones de sensores y movimientos, que van desde 0.614 hasta 0.739. Estos valores representan la fuerza y dirección de la relación entre las señales captadas por los sensores en cada combinación.

Por ejemplo, la combinación 'BD_MT_S1_M1', 'BD_MT_S2_M1', 'BD_MT_S3_M1' tiene una correlación de 0.675, mientras que 'BD_MT_S1_M2', 'BD_MT_S2_M2', 'BD_MT_S3_M2' tiene una correlación de 0.739. Estos valores indican que las señales de los sensores en estas combinaciones tienden a cambiar juntas en cierta medida.

Los valores más bajos de correlación se encuentran en las combinaciones relacionadas con el músculo bíceps superior. Esto sugiere que las señales de los sensores en esta zona tienen una relación más débil entre sí en comparación con las otras combinaciones.

Los detalles completos de las correlaciones para todas las combinaciones se pueden encontrar en las tablas de la prueba en el Anexo G.

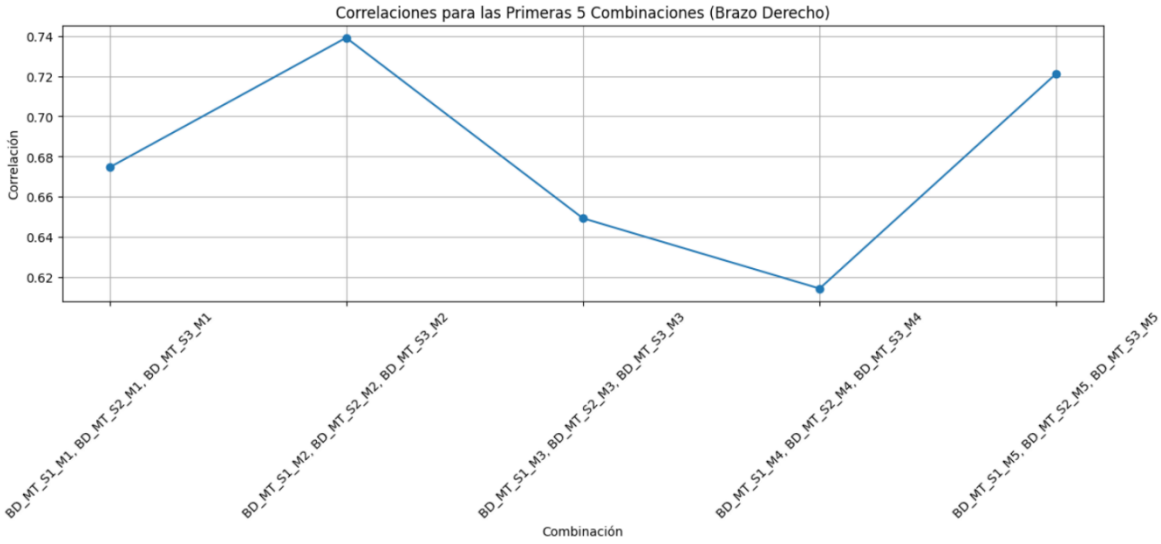


Ilustración 4-6 Análisis de correlaciones para las cinco primeras combinaciones

Realizado por: Vaca-Paguay, 2024.

4.3.2 Análisis de correlación entre dataframes brazo derecho e izquierdo (prueba espejo)

En la prueba de correlación entre los dataframes del brazo derecho e izquierdo, también denominada "prueba espejo", se analiza la similitud en la actividad muscular entre ambos brazos durante los mismos movimientos. Esta prueba es importante porque permite identificar posibles asimetrías en la actividad muscular entre los brazos, lo cual puede ser relevante para comprender el funcionamiento del sistema muscular y para diseñar estrategias de rehabilitación o entrenamiento más efectivas (Fiallos, 2021).

Tabla 4-8 Análisis de correlación (prueba espejo)

N°	Combinaciones	Coeficiente de correlación
1	BI_MT_S1_M1 y BD_MT_S1_M1	-0.067
2	BI_MT_S1_M2 y BD_MT_S1_M2	0.107
3	BI_MT_S1_M3 y BD_MT_S1_M3	0.028
4	BI_MT_S1_M4 y BD_MT_S1_M4	-0.030
5	BI_MT_S1_M5 y BD_MT_S1_M5	0.173

Realizado por: Vaca-Paguay, 2024.

En la Tabla 4-8 se muestran las correlaciones para las 5 primeras combinaciones de sensores entre los brazos derecho e izquierdo. Los coeficientes de correlación, que van desde -0.067 hasta 0.173, indican el grado de relación entre las señales de los sensores en cada combinación es relativamente bajo. Estos valores sugieren que las señales de los sensores en ambos brazos tienen una correlación baja o incluso negativa en algunos casos, esto podría interpretarse como una notable

variación en la actividad muscular y en las señales producidas en los diferentes brazos durante los movimientos analizados.

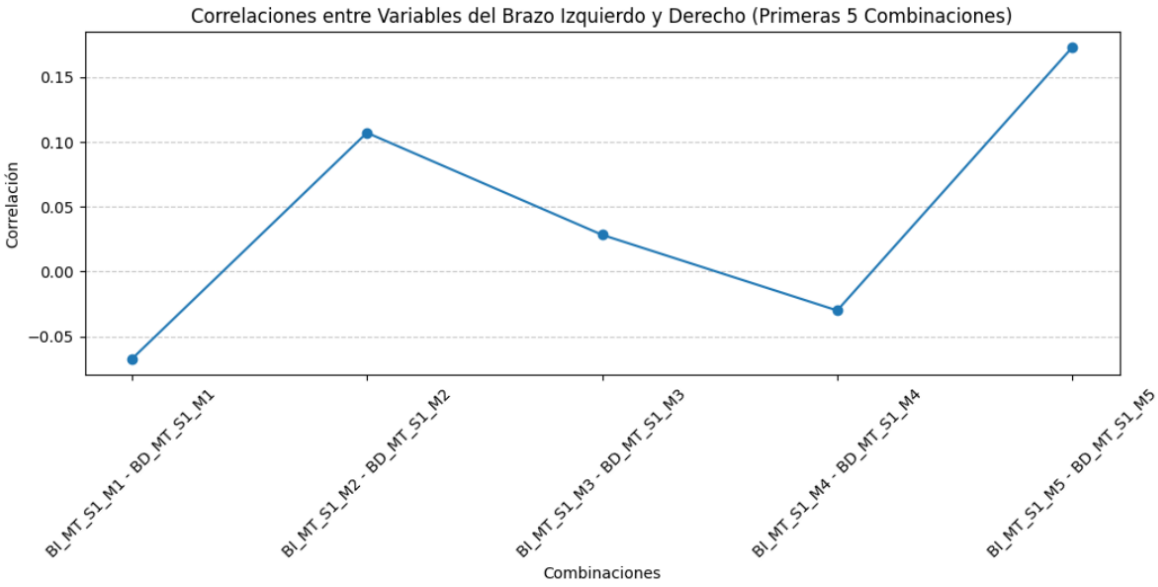


Ilustración 4-7 Correlaciones entre variables del brazo izquierdo y derecho

Realizado por: Vaca-Paguay, 2024.

En la Ilustración 4-7 se observa la gráfica de correlaciones para estas combinaciones. Esta representación visual confirma los bajos niveles de relación entre las combinaciones espejo de los sensores en los brazos derecho e izquierdo.

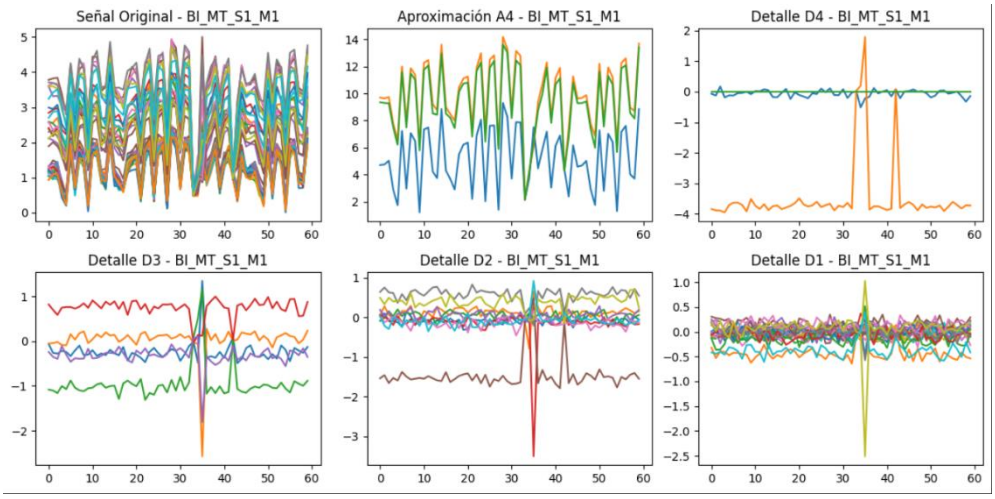
4.3.3 Análisis de ondículas o wavelet

El análisis wavelet es una técnica de procesamiento de señales el cual descompone una señal en diferentes componentes en el dominio de la frecuencia y en el tiempo usando algunas funciones matemáticas conocidas como ondículas (Rodríguez de la Gala Ureña, 2023). Para nuestro análisis en particular usamos la ondícula de Haar. Esta es una de las más simples y se caracteriza por ser una función escalón, es útil para detectar cambios abruptos en una señal, lo que la hace adecuada para el análisis de señales mioeléctricas. La ondícula de Haar se expresa de acuerdo con la Ecuación 6

$$\psi(t) = \begin{cases} 1, \text{si} \quad 0 \leq t < \frac{1}{2} \\ -1, \text{si} \quad \frac{1}{2} \leq t < 1 \\ 0, \quad \text{en otro caso} \end{cases} \tag{6}$$

La función se emplea para calcular los coeficientes de aproximación y detalle de una señal en múltiples niveles de descomposición. En este caso, se realiza una descomposición en 4 niveles, lo que produce 5 conjuntos de coeficientes: A4, D4, D3, D2 y D1.

En la Ilustración 4-9 se puede observar los diferentes niveles de descomposición para las señales del movimiento uno y dos del músculo tensor correspondiente al sensor uno. Para las señales originales se identifican patrones marcados con picos cambiantes entre valores con un rango de 0 a 5. Para las aproximaciones A4, las señales del primer dataframe muestran dos patrones idénticos en escala y forma de onda, junto con una tercera señal que, aunque mantiene la forma, tiene valores más bajos. En contraste, las tres señales del segundo dataframe mantienen su forma y escala. En cuanto a los detalles D4, la señal del primer dataframe muestra una línea continua en 0, la segunda señal tiene modulaciones cercanas a cero, y la tercera señal presenta ondulaciones más pronunciadas con picos en x=35 y x=42. Estos patrones se repiten en los detalles siguientes, con señales cada vez más onduladas y algunos picos en puntos específicos de x (Rodríguez de la Gala Ureña, 2023).



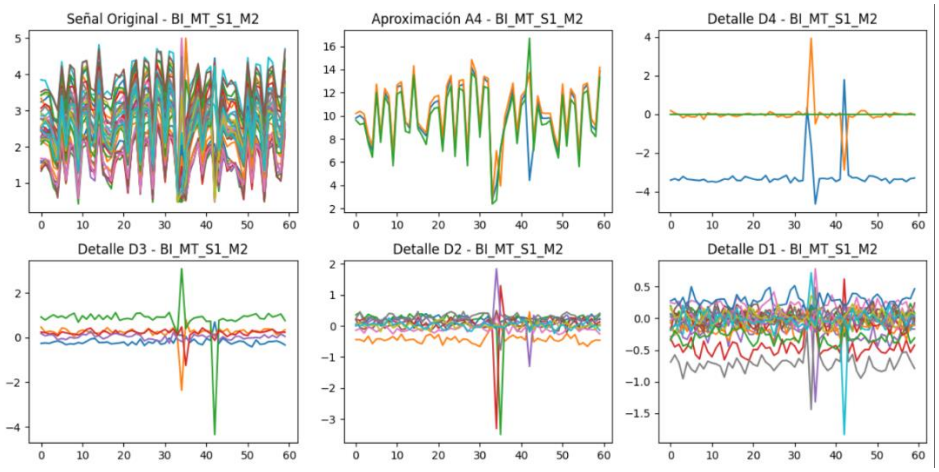


Ilustración 4-8 Análisis wavelet para los dos primeros dataframe

Realizado por: Vaca-Paguay, 2024.

Estos resultados dan a entender que las señales analizadas para cada dataframe presentan características distintivas las cuales indican diferentes comportamientos para cada actividad muscular y zona registrada, lo cual es relevante para distinguir diferentes tipos de movimientos en el análisis de señales mioeléctricas. En el Anexo I se observa el análisis wavelet para los distintos dataframes.

4.4 Resultados del modelo categórico

Para analizar los resultados del modelo categórico es importante considerar la precisión general del modelo en la clasificación de los diferentes tipos de movimientos. En la Tabla 4-9 se observa los resultados del modelo para cada categoría la cual revela un rendimiento en líneas generales bastante alto para la clasificación de los movimientos (Schlotthauer, 2022).

Tabla 4-9 Precisión por categoría

Categoría	Precisión
1	1.00
2	1.00
3	1.00
4	0.986
5	100
6	0.985
7	0.986
8	0.986

Realizado por: Vaca-Paguay, 2024.

Además, en la Ilustración 4-9 se muestra el accuracy (precisión) y val_accuracy (precisión de validación) del modelo. Estos valores proporcionan una medida de la eficacia del modelo en la clasificación de los movimientos.

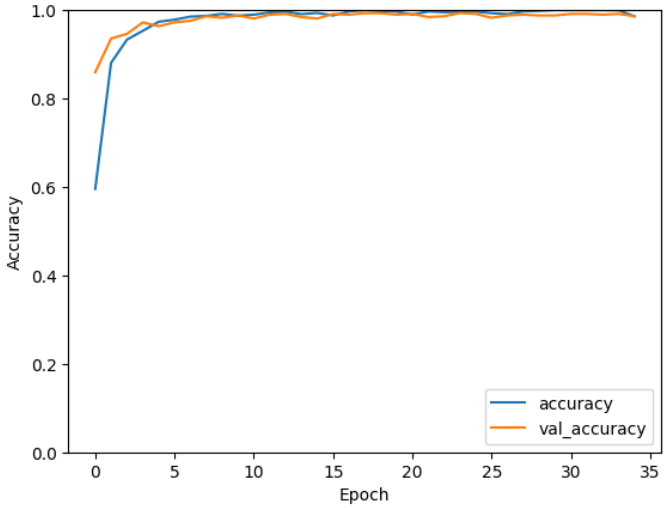


Ilustración 4-9 Precisión del modelo

Realizado por: Vaca-Paguay, 2024.

En la Ilustración 4-10, se observa que las categorías 1, 2, 3 y 5 muestran una precisión absoluta dando un rendimiento del 100% lo que indica que el modelo pudo identificar correctamente todos los movimientos pertenecientes a estas categorías. Para el resto de las categorías la precisión del modelo sigue siendo muy alta con valores muy cercanos al 100%, lo que indica una capacidad robusta del modelo para distinguir entre estos movimientos. Estos resultados destacan la eficacia del modelo para la clasificación de señales mioeléctricas con una alta precisión (Del Carmen & Ramos, 2021).

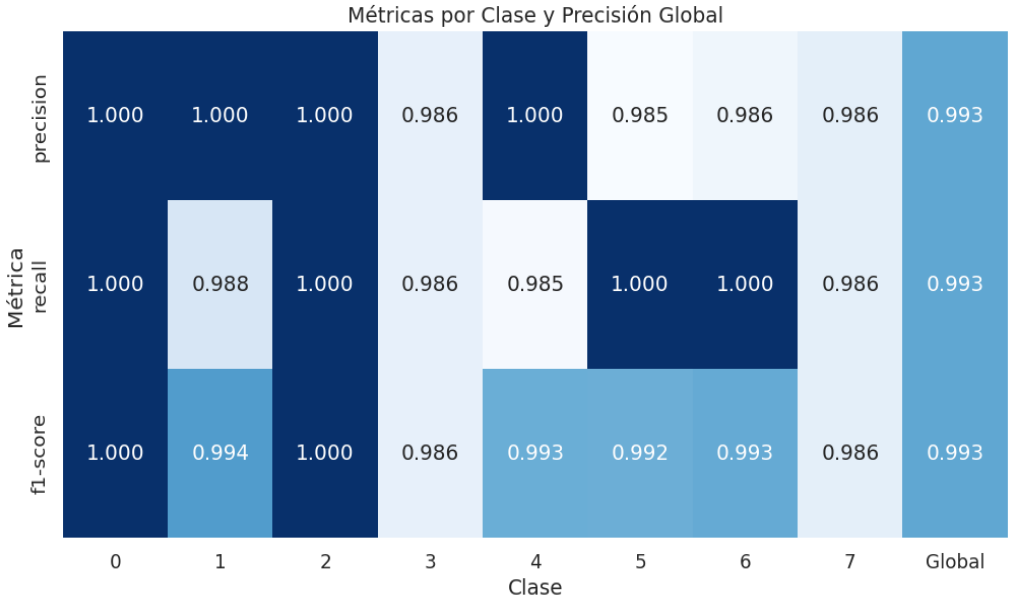


Ilustración 4-10 Métricas obtenidas a partir del modelo categórico

Realizado por: Vaca-Paguay, 2024.

Para poner en funcionamiento el modelo, se ingresaron 30 señales de prueba, las cuales se buscaban categorizar con el modelo. Los resultados de esta prueba se detallan en la Tabla 4-10. El modelo ha logrado una precisión del 100% en la clasificación de las señales mioeléctricas. Cada una de las señales verificadas ha sido correctamente categorizada, lo que indica una alta capacidad del modelo para identificar y clasificar los diferentes tipos de movimientos asociados con las señales. Este resultado es especialmente destacable dado que las señales mioeléctricas pueden ser complejas y variadas, lo que hace que su clasificación sea un desafío (Schlotthauer, 2022).

Tabla 4-10 Resultados de las señales verificadas

Señal	Categoría real	Predicción	Verificación
1	2	2	True
2	5	5	True
3	7	7	True
4	5	5	True
5	4	4	True
6	4	4	True
7	4	4	True
8	7	7	True
9	3	3	True
10	6	6	True
11	8	8	True
12	8	8	True
13	2	2	True
14	5	5	True
15	3	3	True
16	3	3	True
17	2	2	True
18	1	1	True
19	1	1	True
20	3	3	True
21	3	3	True
22	4	4	True
23	4	4	True
24	7	7	True
25	5	5	True
26	5	5	True
27	6	6	True
28	3	3	True
29	5	5	True
30	8	8	True

Realizado por: Vaca-Paguay, 2024.

4.5 Resultados del modelo de predicción temporal

Los resultados del modelo de predicción temporal demuestran su capacidad para completar señales temporales a partir de datos parciales o incompletos. Se observa una correspondencia significativa entre las predicciones del modelo y las señales originales, lo que sugiere un aprendizaje efectivo de las características temporales de las señales mioeléctricas por parte de la red neuronal convolucional (CNN) de predicción temporal.

En la Ilustración 4-9 se muestra el accuracy (precisión) y val_accuracy (precisión de validación) del modelo de predicción temporal.

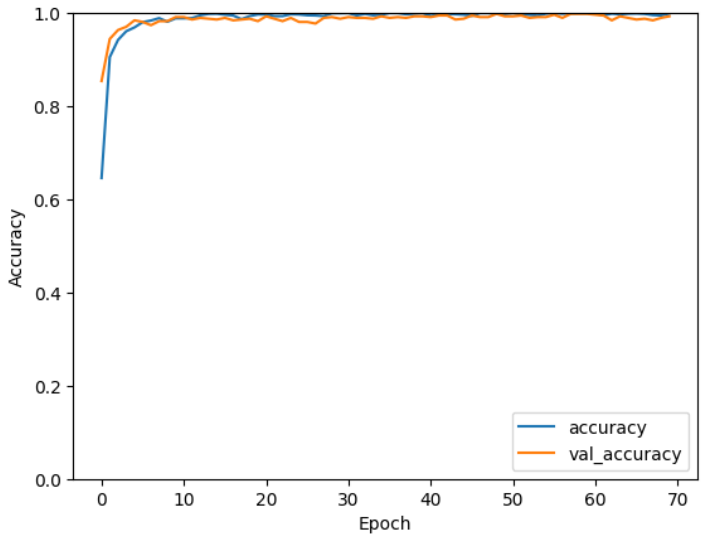


Ilustración 4-11 Precisión del modelo

Realizado por: Vaca-Paguay, 2024.

En las señales con un rizado suave y no muy cambiante, donde la tendencia general de la señal es uniforme, se observa una precisión menor en comparación con las señales que presentan cambios bruscos de escala o comportamientos dinámicos. Esto puede darnos a entender que la red ha aprendido a modelar de manera más efectiva las tendencias lineales en señales con rizado suave, mientras que, en señales más dinámicas, las predicciones del modelo son más precisas al seguir la forma general de la señal original.

En la Tabla 4-11 se muestra la tabla de correlaciones entre la predicción del modelo y la señal original de cada data frame. La tabla completa con todos los datos se ubica en el Anexo J.

Tabla 4-11 Correlaciones entre señales originales y predicciones

Data Frame BI	Correlación	Data Frame BD	Correlación
BI_MBS_S1_M1	0.989	BD_MT_S3_M8	0.982
BI_MBS_S1_M2	0.975	BD_MT_S3_M6	0.988

Realizado por: Vaca-Paguay, 2024.

Estos resultados, respaldados por la Ilustración 4-12, confirman la capacidad del modelo para predecir el comportamiento temporal de las señales mioeléctricas.

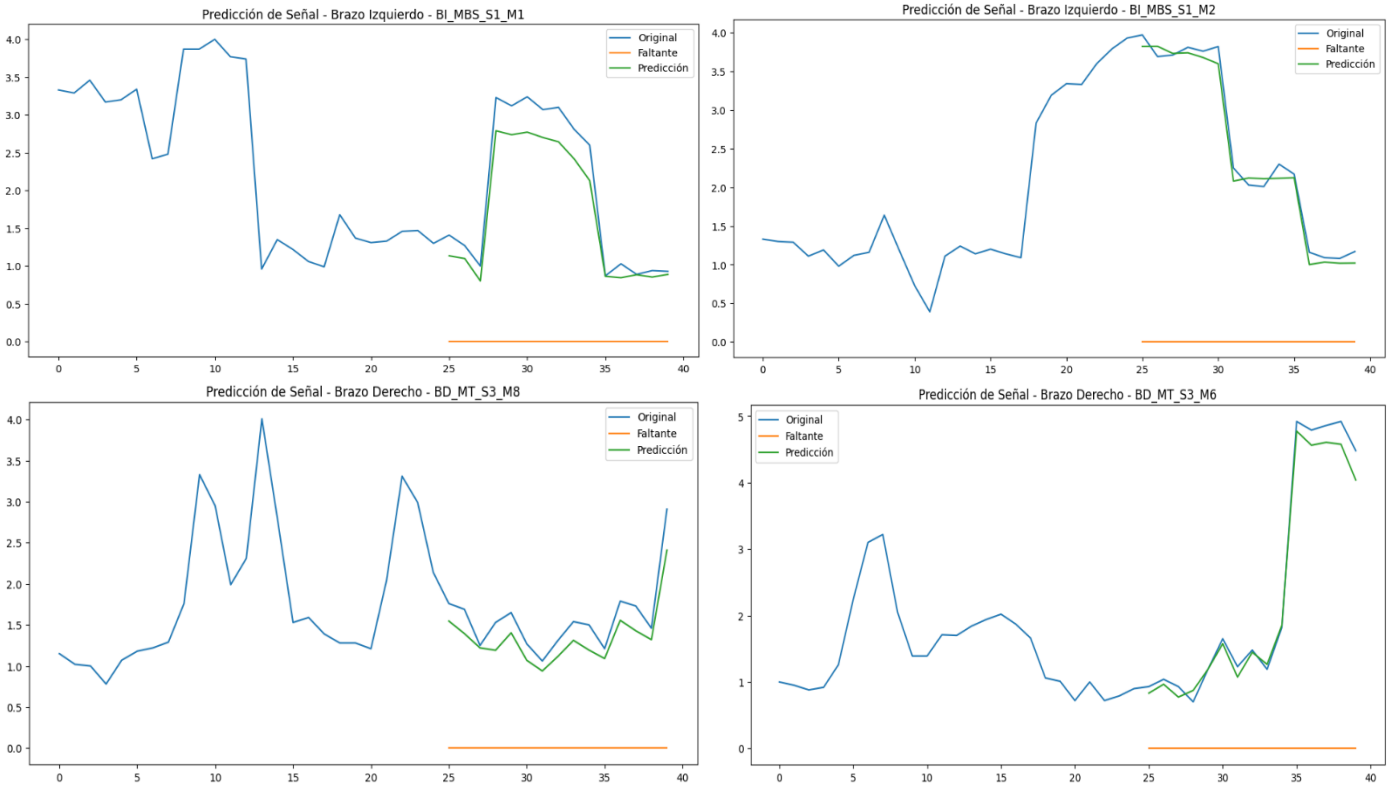


Ilustración 4-12 Predicciones temporales para señales mioeléctricas

Realizado por: Vaca-Paguay, 2024.

En la Ilustración 4-13 se presentan las métricas finales del modelo, destacando un Error Cuadrático Medio (MSE) de 0.0906, un Error Absoluto Medio (MAE) de 0.2094, un coeficiente de determinación (R^2) de 0.9221 y un promedio de residuos de 0.2094. Estos resultados reflejan un rendimiento positivo en la capacidad predictiva del modelo.

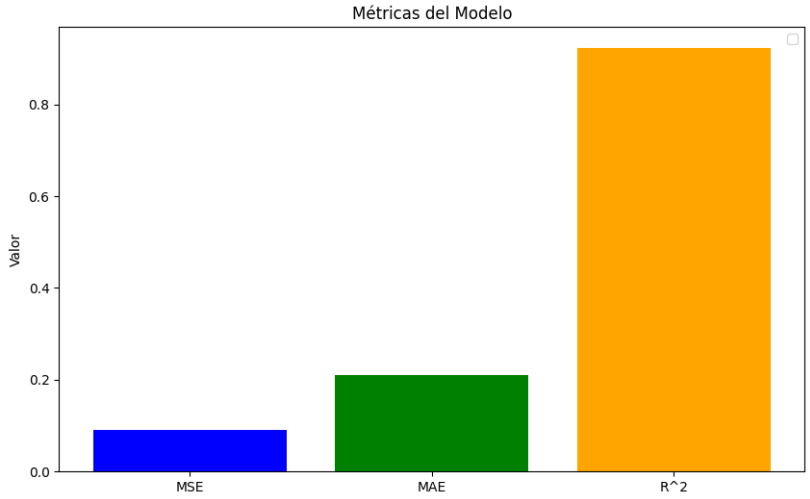


Ilustración 4-13 Métricas del modelo de predicción temporal.

Realizado por: Vaca-Paguay, 2024.

CAPITULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- Se realizó un estudio profundo sobre las señales mioeléctricas, tras esa investigación se logró la comprensión, su generación y sus características. Este análisis permitió no solo saber sobre la complejidad de estas señales, sino también identificar los aspectos clave que deben ser considerados en el diseño y desarrollo de sistemas para su adquisición y procesamiento. Además, se determinó los factores en la señal que pueden intervenir al momento de capturar e interpretar dichas señales.
- Con la investigación realizada se pudo identificar de manera precisa tantos los componentes hardware como softwares necesarios para la adquisición, procesamiento y análisis de las señales mioeléctricas. Y así se evaluó diferentes dispositivos y herramientas disponibles en el mercado, teniendo en cuenta los requisitos como precisión, velocidad de procesamiento, robustez. Lo que llevó a seleccionar la tarjeta de desarrollo ESP32S y el sensor MyoWare como hardware y por otro lado Python con Google Colab para el software. Considerando aspectos como la interfaz con el usuario asegurando la eficacia y viabilidad del diseño propuesto.
- A través de un diseño iterativo para la adquisición de señales mioeléctricas de un grupo de personas, se desarrolló una interfaz dentro del entorno Pycharm con código Python, el cual fue un sistema altamente funcional y adaptable. El diseño integró los componentes hardware y software definidos en el capítulo 3, garantizando la fiabilidad del sistema de adquisición de señales mioeléctricas.
- Utilizando técnicas avanzadas de Machine Learning, se desarrolló un sistema sólido y efectivo basado en redes neuronales convolucionales para interpretar y predecir movimientos en prótesis funcionales basados en señales mioeléctricas. El enfoque se centró en la extracción de las señales, la selección de algoritmos de aprendizajes adecuados y la optimización de los modelos resultantes para garantizar la precisión y fiabilidad del modelo final.
- El sistema en conjunto demostró un rendimiento óptimo. Se realizó la validación del sensor mioeléctrico tras una comparación de sus aspectos teóricos con su desempeño

práctico. Así mismo, se llevó a cabo pruebas de funcionamiento del modelo categórico y del modelo de predicción temporal abarcando diferentes escenarios y condiciones de uso. Los resultados obtenidos de estas pruebas validan la efectividad y utilidad del modelo implementado en entornos prácticos y reales, destacando la relevancia para mejorar la calidad de vida de las personas con amputación y/o malformación transradial que tienen la necesidad de usar prótesis funcionales.

5.2 Recomendaciones

- Ya que las señales mioeléctricas son complejas de trabajar, es fundamental realizar un seguimiento a largo plazo para obtener una retroalimentación continúa para mejorar el rendimiento del sistema con el tiempo y evaluar la aceptación de estas tecnologías desarrolladas en términos de la calidad de vida, la autonomía y la integración social de las personas con amputación y/o malformación transradial.
- Ampliar el conjunto de datos base con el fin de fortalecer la robustez de los modelos. Al aumentar el tamaño del dataset de entrenamiento, se brinda al modelo la oportunidad de comprender e interpretar patrones aún más complejos y variables, mejorando así su capacidad predictiva y su capacidad para adaptarse a una amplia gama de situaciones.
- En este proyecto se utilizó componentes de hardware con un coste relativamente bajo, como lo son los sensores MyoWare, el cual presento ciertos desafíos al momento de la colocación en los pacientes, por lo cual, se debe considerar que los sensores tienen limitaciones en términos de precisión y eficiencia. Por lo tanto, se sugiere la incorporación de tecnologías más sofisticadas y actualizadas para garantizar una captura más precisa de las señales mioeléctricas y una mejora significativa en la funcionalidad del sistema.
- Para validar la efectividad y utilidad clínica de los modelos, se recomienda seguir con la etapa de construcción de prótesis de activación mioeléctrica para realizar pruebas en entornos clínicos reales. Esta validación clínica ayuda a adaptar los modelos a las necesidades reales.
- Es muy importante implementar medidas que reduzcan interrupciones y variables externas durante la adquisición de señales para evitar datos atípicos, mediante protocolos de adquisición y la identificación temprana de posibles fuentes de interferencia, asegurando la calidad y consistencia de los datos recolectados.
- Emplear técnicas de optimización de hiperparámetros, como la búsqueda aleatoria o búsqueda en cuadricula, para encontrar la combinación óptima de parámetros de los modelos.

BIBLIOGRAFÍA

AGUAGALLO, F. (2019). *IMPLEMENTACIÓN DE UN SISTEMA DE ENTRENAMIENTO PARA MEDICIÓN Y ANÁLISIS DE SEÑALES ELÉCTRICAS Y CARACTERIZACIÓN DE SENSORES E INSTRUMENTOS PARA EL LABORATORIO DE ELECTRÓNICA DE LA FACULTAD DE MECÁNICA.*

ALBA, D., & CALLE, D. (2020). *APLICACIÓN DE TÉCNICAS DE MACHINE LEARNING BASADO EN INFORMACIÓN SÍSMICA PARA PROFUNDIZAR LA PROBABILIDAD DE TERREMOTOS MEDIANTE EL USO DE REGRESIÓN LOGÍSTICA Y REDES NEURONALES.* Obtenido de UNIVERSIDAD DE GUAYAQUIL: https://repositorio.ug.edu.ec/server/api/core/bitstreams/68a85f0a-1530-4a86-a5fb-5eadc3bf7550/content

ALJURE JIMÉNEZ, Y., et. al. (2021). *Clasificación de Flores con Redes Neuronales Convolucionales*. www.udea.edu.co

AZPILCUETA, R. (2020). *Dispositivo de adquisición de señales mioeléctricas*. http://rinfi.fi.mdp.edu.arhttp://rinfi.fi.mdp.edu.ar/xmlui/handle/123456789/456

BARAHONA, M., & JARAMILLO, L. (2022). *REGRESIÓN Y CARACTERIZACIÓN DE SEÑALES ELECTROENCEFALOGRÁFICAS EMPLEANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL MARÍA ALEJANDRA BARAHONA GARCÍA REGRESION Y CARACTERIZACIÓN DE SEÑALES ELECTROENCEFALOGRÁFICAS EMPLEANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL*.

BARCO, R. (2013). *Discretización de Señales.* Obtenido de Universidad Nacional del Sur: http://lcr.uns.edu.ar/fvc/NotasDeAplicacion/FVC-RodrigoBarco.pdf

BERRIOS GÓMEZ, S., & RIVERA HERRERA, H. (2022). Sistema IoT basado en ESP32 para el control y monitoreo de cultivos en invernadero con enfoque de agricultura 4.0. *Ingeniería investiga*.

BODERO, E. M., et. al. (2019). Google Colaboratory como alternativa para el procesamiento de una red neuronal convolucional. In *ISSN* (Bd. 41).

BRAVO, C. (2018). *MONITOREO INALÁMBRICO DE CONSUMO ELÉCTRICO CON SENSOR NO INVASIVO Y MICROCONTROLADOR ARDUINO.* Obtenido de INSTITUTO TECNOLÓGICO DE CD. GUZMÁN: https://rinacional.tecnm.mx/bitstream/TecNM/1703/1/Tesis%20CE%CC%81SAR%20NICOLA%CC%81S%20BRAVO%20DI%CC%81AZ.pdf

BRAVO LOOR, S., et. al. (2023). Características psicosociales del alumnado ecuatoriano con discapacidad motora. *Telos Revista de Estudios Interdisciplinarios en Ciencias Sociales*, *25*(1), 103–120. https://doi.org/10.36390/telos251.08

BRUSIL, C. (2020). *ANÁLISIS COMPARATIVO ENTRE APRENDIZAJE SUPERVISADOY APRENDIZAJE SEMI-SUPERVISADO PARA LA.* Obtenido de ESCUELA POLITÉCNICA NACIONAL: https://bibdigital.epn.edu.ec/bitstream/15000/20723/1/CD%2010239.pdf

CARO ZAPATA, S. (2022). *Verificación de la integridad de los datos adquiridos por un sistema de medición de variables cinéticas, cinemáticas y fisiológicas, ajustado a una bicicleta horizontal en conexión con un juego serio.*

CASTRO, M. C. F., et. al. (2022). A Hybrid 3D Printed Hand Prosthesis Prototype Based on sEMG and a Fully Embedded Computer Vision System. *Frontiers in Neurorobotics*, *15*. https://doi.org/10.3389/fnbot.2021.751282

CHEN, Z., et. al. (2019). Multivariate Gaussian and Student-t process regression for multioutput prediction.

CIFUENTES, I. (2012). Extracción de Características y Clasificación de Señales Electromiográficas Utilizando la Transformada Hilbert-Huang y Redes Neuronales. *©INAOE*, *1*.

CRAWFORD, L. E., & VAVRA, D. T. (2016). *UR Scholarship Repository Typing with EMG using MyoWare*. https://learn.sparkfun.com/tutorials/how-to-solder---

COBO, M., & LLORET IGLESIAS, L. (2023). *Inteligencia artificial y medicina.* Madrid: CSIC.

DALCAME. (2024). *Temperatura Corporal*. Obtenido de Grupo de Investigación Biomédica: https://www.dalcame.com/tc.html

DEL BRÍO, A. (2020). *Tecnología electrónica en Ingeniería biomédica*.

DEL CARMEN, S., & RAMOS, B. (2021). *Modelado y generación de patrones de señales eléctricas para estudios de propagación en tejidos biológicos mediante simulación.*

DUFOUR, M., & DEL VALLE , S. (2020). *Los Músculos. Anatomía Clínica de las Extremidades.* Paidotribo.

ESAA, R. R., et. al. (2023). Short-term hand gestures recognition based on electromyography signals. *IAES International Journal of Artificial Intelligence*, *12*(4), 1765–1773. https://doi.org/10.11591/ijai.v12.i4.pp1765-1773

FERNÁNDEZ, A. (2022). *ELABORACIÓN DE UN ATLAS DE IMÁGENES HISTOLÓGICAS TEJIDO MUSCULAR.* Obtenido de Universidad de Valladolid: https://uvadoc.uva.es/bitstream/handle/10324/54122/TFG-H2435.pdf?sequence=1&isAllowed=y

FERREIRA-ARQUEZ, H. (2020). *CABEZA EXTRA DEL MUSCULO BICEPS BRACHII.*

FIALLOS, G. (2021). La Correlación de Pearson y el proceso de regresión por el Método de Mínimos Cuadrados. *Ciencia Latina Revista Científica Multidisciplinar*, *5*(3), 2491–2509. https://doi.org/10.37811/cl_rcm.v5i3.466

FUENTES, J. (2021). *Estudio y diseño de materiales de impresión 3D que soporten los sistemas de esterilización médicos.* Obtenido de UNIVERSIDAD POLITÉCNICA DE VALENCIA: https://riunet.upv.es/bitstream/handle/10251/180857/Fuentes%20-%20Estudio%20y%20diseno%20de%20materiales%20de%20impresion%203d%20que%20soporten%20los%20sistemas%20de%20esterilizac....pdf?sequence=1&isAllowed=y

GARNICA, D. A. B., et. al. (2020). *Desarrollo de un sistema de control para el movimiento de un prototipo de prótesis de mano a partir del reconocimiento de señales Mioeléctricas*.

GIGLI, A., et. al. (2023). Progressive unsupervised control of myoelectric upper limbs. *Journal of Neural Engineering*, *20*(6). https://doi.org/10.1088/1741-2552/ad0754

HE, X., et. al. (2022). Localization of the Centre of the Highest Region of the Triceps Brachii Muscle Spindle Abundance and its Significance in Muscle Spasticity Block. In *Int. J. Morphol* (Bd. 40, Nummer 4).

HERMITAÑO, J. (2022). *Aplicación de Machine Learning en la Gestión de Riesgo de Crédito Financiero_ Una revisión sistemática*.

HERNÁNDEZ MONTEJANO, C. O. (2022). *Modelo de clasificación de señales de electrorretinograma para diagnóstico de retinopatía diabética temprana*.

IKISS, J. (2020). *SISTEMA DE ADQUISICIÓN DE DATOS CON ESP32*.

JÁCOME DEL VALLE, D. M. (2022). *Diseño e Implementación de un Robot Seguidor de Línea NatCar Utilizando Sensores QTR 1A y un Módulo ESP 32*. INSTITUTO SUPERIOR TECNOLÓGICO VIDA NUEVA.

JOLLES, J. W. (2021). Broad-scale applications of the Raspberry Pi: A review and guide for biologists. In *Methods in Ecology and Evolution* (Bd. 12, Nummer 9, S. 1562–1579). British Ecological Society. https://doi.org/10.1111/2041-210X.13652

KERVER, N., et. al. (2023). The multi-grip and standard myoelectric hand prosthesis compared: does the multi-grip hand live up to its promise? *Journal of NeuroEngineering and Rehabilitation*, *20*(1). https://doi.org/10.1186/s12984-023-01131-w

LINARES, A., & ROSAS, D. (2019). Desarrollo de prótesis electromecánica de miembro superior. *3*(10 23-30).

LUQUE, R. (2021). En R. Luque, *Introducción a la anatomía* (pág. 262). Bogotá: Editorial Universidad del Rosario.

MADOU, R., et. al. (2020). Señales bioeléctricas del cuerpo: de la ingeniería electrónica a la performance artística. *¡ Cuerpo, Máquina, Acción!*, *4*.

MICHAUD, K. (2022). *COMPARACIÓN DE LA REALIDAD VIRTUAL CON LA TERAPIA ESPEJO EN LA REHABILITACIÓN DE PACIENTES QUE SUFREN DOLOR EN MIEMBRO FANTASMA EN LA EXTREMIDAD SUPERIOR:*

MOLER, C., & LITTLE, J. (2020). A history of MATLAB. *Proceedings of the ACM on Programming Languages*, *4*(HOPL). https://doi.org/10.1145/3386331

NAVARRO, C. E. (2021). *Implementación de un generador de señales de circuitos electrónicos elaborados para el mejoramiento del proceso enseñanza aprendizaje en la asignatura de electrónica y robótica de la carrera de Tecnología de la Información.* Universidad Estatal del Sur de Manabí.

NESCOLARDE LÓPEZ, D. (2023). *Diseño e implementación de un dispositivo IoT de bajo coste para la medida de la calidad del aire*. UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE.

PADILLA, L., & NICOLALDE, E. M. (2022). MONETARY INTEGRATION IN SOUTH AMERICA: ELECTION OF CANDIDATES THROUGH UNSUPERVISED MACHINE LEARNING. *Revista de Economia Mundial*, *2022*(61), 63–89. https://doi.org/10.33776/rem.v0i61.5155

PAGÁN GONZÁLEZ, G. (2023). *Sistema de adquisición de datos de consumo eléctrico por Modbus y análisis exploratorio en Python.*

PÉREZ, E. (2019). *ALGORITMO DE RANDOM FOREST APLICADO A LA DETECCIÓN DE FRAUDE EN EL SISTEMA BANCARIO ECUATORIANO.* . Obtenido de ESCUELA POLITÉCNICA NACIONAL: https://bibdigital.epn.edu.ec/bitstream/15000/20592/1/CD%2010099.pdf

QUEVEDO, M. A. C., & GOMEZ DONOSO, F. (2021). *Control mediante aprendizaje por refuerzo del robot Pepper*.

QUINTERO ÁVALO, J. D. (2023). *Estación Meteorológica de bajo costo Estudio de KiCad como Herramienta de Diseño Electrónico para Mejorar la Eficiencia en Sistemas de Riego*.

RAMÍREZ, W., & RAMÍREZ, C. (2023). *Programación de Inteligencia Artificial. Curso Práctico.* Ra-Ma S.A. Editorial y Publicaciones.

RATHIKA, P. D., ET. AL. (2021). *Gesture Based Robot Arm Control*.

RODRÍGUEZ, A. (2020). *IMPACTO DEL USO DE UN SILLÍN DE BICICLETA INESTABLE EN LA EFICIENCIA ENERGÉTICA Y LA ACTIVIDAD MUSCULAR EN UNA PRUEBA DE EJERCICIO SUBMÁXIMO EN SUJETOS SANOS*. UNIVERSIDAD DE CHILE.

RODRÍGUEZ DE LA GALA UREÑA, C. A. (2023). *Análisis de ondículas para señales de EEG en el habla imaginada*.

ROZO-GARCÍA, F. (2020). Revisión de las tecnologías presentes en la industria 4.0. *Revista UIS Ingenierías*, *19*(2), 177–191. https://doi.org/10.18273/revuin.v19n2-2020019

SADRIDDINOVICH JALOLOV, T. (2023). Solving Complex Problems in Python. *AMERICAN Journal of Language, Literacy and Learning in STEM Education*.

SALAMANCA, I., & CASTRO, E. (2020). *Técnicas de aprendizaje automático aplicadas en los sistemas de predicción*.

SÁNCHEZ GRANJA, G. (2021). *Implementación de algoritmo de machine learning en un sistema embebido para control de prótesis activa utilizando señales mioeléctricas*.

SANDEEP RAO, T., et. al. (2021). *ESP32 Based Implementation of Water Quality and Quantity Regulating System*.

SCHLOTTHAUER, G. (2022). *Procesamiento, análisis y modelado de señales biomédicas: un enfoque integrador*.

SHAKIROVICH ISMAILOV, A. (2022). *Study of arduino microcontroller board*. www.openscience.uz

SIDDIQ AHMED, S., et. al. (2021). Design and multichannel electromyography system-based neural network control of a low-cost myoelectric prosthesis hand. *Mechanical Sciences*, *12*(1), 69–83. https://doi.org/10.5194/ms-12-69-2021

SIMON, A. M., et. al. (2023). Myoelectric prosthesis hand grasp control following targeted muscle reinnervation in individuals with transradial amputation. *PLoS ONE*, *18*(1 January). https://doi.org/10.1371/journal.pone.0280210

SOLANET, M. (2021). *INTELIGENCIA ARTIFICIAL UNA MIRADA MULTIDISCIPLINARIA.* Buenos Aires: Academia Nacional de Ciencias Morales y Políticas.

SUÁREZ GARCÍA, M. (2021). *Adaptación de la Prótesis de Mano Basada en Soft-Robotics PrExHand para la Evaluación con Usuarios no Patológicos en Pruebas Funcionales*.

SUBERVIOLA, A. (2019). *CONTROL DE UN EXOESQUELETO MEDIANTE SEÑALES EMG (ELECTROMIOGRÁFICAS).* Obtenido de Universidad del país vasco Euskal Herriko Unibertsitatea: https://www.ehu.eus/ccwintco/uploads/6/67/Tesis_doctoral_suberviola_v13_(final).pdf

TENE, M., & TENE, R. (2022). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MICROCONTROLADOR PARA EL MONITOREO DE MÁQUINAS ROTATIVAS*. ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO.

VASCONCELLOS, F. (2022). *PROCESAMIENTO DE SEÑALES EEG BASADO EN PYTHON*.

VASQUEZ, A. (2021). *Medición de la Saturación de Oxigeno Muscular y su Implicación en la Fatiga, Rendimiento y la Salud.* Obtenido de Universidad de Extremadura: https://www.educacion.gob.es/teseo/imprimirFicheroTesis.do?idFichero=K%2BJeZL8F6%2FA%3D

VÁZQUEZ, A., et. al. (2020). ADQUISICIÓN Y FILTRADO DE SEÑALES EMGS EMGS SIGNAL ACQUISITION AND FILTERING [Tecnológico Nacional de México en Celaya]. In

Tecnológico Nacional de México en Celaya Pistas Educativas (Bd. 2020, Nummer 137). http://itcelaya.edu.mx/ojs/index.php/pistas

VEL TECH RANGARAJAN SAGUNTHALA, RS. R., & MAHATO, R. (2021). *3D PCB Designing of Protecting Circuit Using Fusion 360*. https://circuitdigest.com

VERA REMACHE, S. L. (2021). *DISEÑO Y CONSTRUCCIÓN DE UN MUÑÓN QUE SIMULE UNA AMPUTACIÓN TRANSFEMORAL*.

VON CHAMIER, L., et. al. (2021). Democratising deep learning for microscopy with ZeroCostDL4Mic. *Nature Communications*, *12*(1). https://doi.org/10.1038/s41467-021-22518-0

XU, F. F., et. al. (2022). In-IDE Code Generation from Natural Language: Promise and Challenges. *ACM Transactions on Software Engineering and Methodology*, *31*(2). https://doi.org/10.1145/3487569

ANEXOS

ANEXO A: CÓDIGON EN ARDUINO – ESP32S

```
const int Sensor1 = 36;  // Pin GPIO 36 Ch1
const int Sensor2 = 39;  // Pin GPIO 39 Ch2
const int Sensor3 = 34; // Pin GPIO 34 Ch3

unsigned long previousMillis = 0;
const int interval = 100;  // Tiempo de espera

void setup() {
  pinMode(Sensor1, INPUT);
  pinMode(Sensor2, INPUT);
  pinMode(Sensor3, INPUT);

  // Detener la comunicación serial
  Serial.end();

  // Reconfigurar la comunicación serial con 115200 baudios y 1 bit de parada
  Serial.begin(115200, SERIAL_8N1);
}

void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    int S1 = analogRead(Sensor1);
    float V1 = (S1 / 4095.0) * 5.0;
    int S2 = analogRead(Sensor2);
    float V2 = (S2 / 4095.0) * 5.0;
    int S3 = analogRead(Sensor3);
    float V3 = (S3 / 4095.0) * 5.0;

    String data = String(V1) + "," + String(V2) + "," + String(V3) + "\n";
    Serial.print(data);

    previousMillis = currentMillis;
  }
}
```

ANEXO B: CÓDIGO PYTHON - INTERFAZ GRÁFICA - ADQUISICIÓN DE SEÑALES

```python
# Muscular Interface - Modeling muscular signals using Machine Learning
# ESPOCH - Electrónica y Automatización 2023
# Version 0.2.1
from datetime import datetime
import os
import time
import serial
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

matplotlib.use("TkAgg")


2 usages
class MuscularSignal:
    def __init__(self, window_):
        self.window = window_
        self.bauds, self.ini_time, self.ini_muscle, self.state, self.muscle, self.file = None, None, None, 0, 0, []
        self.figure_canvas, self.port, self.reference, self.path_, self.t_diff = [], None, False, [], 0
        self.finish, self.data_f, self.path_f, self.actions_c, self.muscle_c = False, [], None, False, False
        self.ch_1 = []

    1 usage
    def update_dir(self, path):
        path_s = path.split('/')
        cad, self.path_f = len(path_s), path_s[0]
        for p in range(1, cad):
            self.path_f += '\\' + path_s[p]
        return self.path_f

    1 usage
    def draw_figure(self, canvas, figure):
        self.figure_canvas = FigureCanvasTkAgg(figure, canvas)
        self.figure_canvas.draw()
        self.figure_canvas.get_tk_widget().pack(side="top", fill="both", expand=1)
        return self.figure_canvas

    1 usage
    def serial_test(self, port_n, bauds, stop):
        try:
            self.bauds = bauds
            serial_stop = serial.STOPBITS_ONE if stop == 1 else serial.STOPBITS_TWO
            self.port = serial.Serial(port=port_n,
                                      baudrate=bauds,
                                      bytesize=serial.EIGHTBITS,
                                      parity=serial.PARITY_NONE,
                                      stopbits=serial_stop)
            print('---------------------------------------')
            print('           Port: ' + self.port.name)
            print('---------- Open port successfully --------')
            self.reference = True
        except serial.SerialException:
            print('------- Port is not available ---------')
            self.reference = False
        return self.reference

    2 usages
    def diff_time(self, t_ini, t_end):
        self.t_diff = (t_end - t_ini).total_seconds()
        return self.t_diff

    1 usage
    def activate_actions(self, act_ctr):
        if act_ctr == 1:
            self.window['_CBM_'].Update(value=True)
        elif act_ctr == 2:
            self.window['_LVB_'].Update(value=True)
        elif act_ctr == 3:
            self.window['_BJB_'].Update(value=True)
        elif act_ctr == 4:
            self.window['_TMO_'].Update(value=True)
        elif act_ctr == 5:
            self.window['_LVO_'].Update(value=True)
        elif act_ctr == 6:
            self.window['_BJO_'].Update(value=True)
        elif act_ctr == 7:
            self.window['_SLO_'].Update(value=True)

    1 usage
    def deactivate_actions(self,):
        self.window['_CBM_'].Update(value=False)
        self.window['_LVB_'].Update(value=False)
        self.window['_BJB_'].Update(value=False)
        self.window['_TMO_'].Update(value=False)
        self.window['_LVO_'].Update(value=False)
        self.window['_BJO_'].Update(value=False)
        self.window['_SLO_'].Update(value=False)

    1 usage
    def activate_muscles(self, mus_ctr):
        if mus_ctr == 1:
            self.window['_BIN_'].Update(value=True)
        elif mus_ctr == 2:
            self.window['_BSU_'].Update(value=True)
        elif mus_ctr == 3:
            self.window['_HMB_'].Update(value=True)

    1 usage
    def deactivate_muscles(self,):
        self.window['_BIN_'].Update(value=False)
        self.window['_BSU_'].Update(value=False)
        self.window['_HMB_'].Update(value=False)

    def save_signal(self, time_sg, channel_1):
        if len(self.ch_1) < 1:
            self.ini_time = time.time()
            self.rest_sg = 0
        if self.rest_sg < time_sg:
            self.rest_sg = time.time() - self.ini_time
            print(self.rest_sg)
            self.ch_1.append(channel_1[-1])
        else:
            self.ch_1 = []

    1 usage
    def read_port(self, time_ac, time_sg, rep, rep_ctr, actions, act_ctr, time_muscle, muscles, mus_ctr,
                  channel_1, channel_2, channel_3, ch_1, ch_2, ch_3, channel_1f, channel_2f, channel_3f, channel_1_p,
                  channel_2_p, channel_3_p):
        if self.actions_c or self.muscle_c or self.ini_time is None:
            self.ini_time = datetime.now()
            self.actions_c, self.muscle_c = False, False
        # read serial port
        signal = self.port.readline()
        print(signal)
        values = signal.split(b",")
        f_values = np.array(list(map(float, values)))
        channel_1.append(f_values[0])
        channel_2.append(f_values[1])
        channel_3.append(f_values[2])

        if mus_ctr < muscles:
            if act_ctr < actions and self.muscle == 0:
                if rep_ctr < rep:
                    self.window['_RPM_'].update(rep_ctr+1)
                    if self.reference:
                        now_time = datetime.now()
                        time_sleep = self.diff_time(self.ini_time, now_time)
                        if self.state == 0:
                            rest_time = np.round(time_ac - time_sleep,  decimals: 2)
                            print('Tiempo espera acción: ' + str(time_ac) + ' ----->> ' + str(rest_time))
                            self.window['_TME_'].update(rest_time)
```

ANEXO C: CÓDIGO PYTHON – GOOGLE COLAB – CLASIFICACIÓN – NORMALIZACIÓN – ENTRENAMIENTO VIRTUAL – MODELOS

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

MODELADO DE LAS SEÑALES MIOELÉCTRICAS PRODUCIDAS POR LAS EXTREMIDADES SUPERIORES, USANDO MACHINE LEARNING, PARA PERSONAS CON NIVELES DE AMPUTACIÓN Y/O MALFORMACIÓN TRANSRADIAL.

Librerías

```
#IMPORTACIÓN DE LIBRERÍAS

import tensorflow as tf
import numpy as np
import pandas as pd
import os
import math
import statsmodels.api as sm
import pywt
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Input, Concatenate, Dense, LSTM
from sklearn.preprocessing import MinMaxScaler, StandardScaler, LabelEncoder
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from google.colab import drive
from itertools import combinations
from sklearn.metrics import accuracy_score
```

Clasificación de archivos

```
#Enlaza el entorno con la nube
drive.mount('/content/drive')
# Rutas de los archivos CSV en tu unidad de Google Drive
carpeta = '/content/drive/My Drive/Signal3.0'

datos_excel_bi_s1 = [f'BI_S1_{i}.csv' for i in range(1, 21)]
datos_excel_bi_s2 = [f'BI_S2_{i}.csv' for i in range(1, 21)]
datos_excel_bi_s3 = [f'BI_S3_{i}.csv' for i in range(1, 21)]

datos_excel_bd_s1 = [f'BD_S1_{i}.csv' for i in range(1, 21)]
datos_excel_bd_s2 = [f'BD_S2_{i}.csv' for i in range(1, 21)]
datos_excel_bd_s3 = [f'BD_S3_{i}.csv' for i in range(1, 21)]

ruta_bi_s1 = [os.path.join(carpeta, archivo) for archivo in datos_excel_bi_s1]
ruta_bi_s2 = [os.path.join(carpeta, archivo) for archivo in datos_excel_bi_s2]
ruta_bi_s3 = [os.path.join(carpeta, archivo) for archivo in datos_excel_bi_s3]
ruta_bd_s1 = [os.path.join(carpeta, archivo) for archivo in datos_excel_bd_s1]
ruta_bd_s2 = [os.path.join(carpeta, archivo) for archivo in datos_excel_bd_s2]
ruta_bd_s3 = [os.path.join(carpeta, archivo) for archivo in datos_excel_bd_s3]

# Lista para almacenar los DataFrames para Brazo Izquierdo
dataframes_Bi_S1 = [[] for _ in range(25)]
dataframes_Bi_S2 = [[] for _ in range(25)]
dataframes_Bi_S3 = [[] for _ in range(25)]

# Lista para almacenar los DataFrames para Brazo Derecho
dataframes_Bd_S1 = [[] for _ in range(25)]
dataframes_Bd_S2 = [[] for _ in range(25)]
dataframes_Bd_S3 = [[] for _ in range(25)]

# Lee archivos CSV para Brazo Izquierdo
for archivos_por_sujeto, dataframes_Bi_S in zip([ruta_bi_s1, ruta_bi_s2, ruta_bi_s3], [dataframes_Bi_S1, dataframes_Bi_S2, dataframes_Bi_S3]):
    for archivos in archivos_por_sujeto:
        datos = pd.read_csv(archivos)
        for fila in range(25):
            if len(datos) > fila:
                fila_original = datos.iloc[fila].values
```

```
                # Divide la fila en bloques de 40 columnas
                bloques = [fila_original[i:i + 40] for i in range(0, len(fila_original), 40)]
                for bloque in bloques:
                    dataframes_Bi_S[fila].append(pd.DataFrame([bloque]))

# Lee archivos CSV para Brazo Derecho
for archivos_por_sujeto, dataframes_Bd_S in zip([ruta_bd_s1, ruta_bd_s2, ruta_bd_s3], [dataframes_Bd_S1, dataframes_Bd_S2, dataframes_Bd_S3]):
    for archivos in archivos_por_sujeto:
        datos = pd.read_csv(archivos)
        for fila in range(25):
            if len(datos) > fila:
                fila_original = datos.iloc[fila].values
                # Divide la fila en bloques de 40 columnas
                bloques = [fila_original[i:i + 40] for i in range(0, len(fila_original), 40)]
                for bloque in bloques:
                    dataframes_Bd_S[fila].append(pd.DataFrame([bloque]))

#  Se Almacena los DataFrames brazo izquierdo
BI_MT_S1_M1, BI_MT_S1_M2, BI_MT_S1_M3, BI_MT_S1_M4, BI_MT_S1_M5, BI_MT_S1_M6, BI_MT_S1_M7, BI_MT_S1_M8, BI_MBI_S1_M1, BI_MBI_S1_M2, BI_MBI_S1_
    pd.concat(dataframes_Bi_S1[i], ignore_index=True) for i in range(24)]
BI_MT_S2_M1, BI_MT_S2_M2, BI_MT_S2_M3, BI_MT_S2_M4, BI_MT_S2_M5, BI_MT_S2_M6, BI_MT_S2_M7, BI_MT_S2_M8, BI_MBI_S2_M1, BI_MBI_S2_M2, BI_MBI_S2_
    pd.concat(dataframes_Bi_S2[i], ignore_index=True) for i in range(24)]
BI_MT_S3_M1, BI_MT_S3_M2, BI_MT_S3_M3, BI_MT_S3_M4, BI_MT_S3_M5, BI_MT_S3_M6, BI_MT_S3_M7, BI_MT_S3_M8, BI_MBI_S3_M1, BI_MBI_S3_M2, BI_MBI_S3_
    pd.concat(dataframes_Bi_S3[i], ignore_index=True) for i in range(24)]

# Se Almacena los DataFrames brazo derecho
BD_MT_S1_M1, BD_MT_S1_M2, BD_MT_S1_M3, BD_MT_S1_M4, BD_MT_S1_M5, BD_MT_S1_M6, BD_MT_S1_M7, BD_MT_S1_M8, BD_MBI_S1_M1, BD_MBI_S1_M2, BD_MBI_S1_
    pd.concat(dataframes_Bd_S1[i], ignore_index=True) for i in range(24)]
BD_MT_S2_M1, BD_MT_S2_M2, BD_MT_S2_M3, BD_MT_S2_M4, BD_MT_S2_M5, BD_MT_S2_M6, BD_MT_S2_M7, BD_MT_S2_M8, BD_MBI_S2_M1, BD_MBI_S2_M2, BD_MBI_S2_
    pd.concat(dataframes_Bd_S2[i], ignore_index=True) for i in range(24)]
BD_MT_S3_M1, BD_MT_S3_M2, BD_MT_S3_M3, BD_MT_S3_M4, BD_MT_S3_M5, BD_MT_S3_M6, BD_MT_S3_M7, BD_MT_S3_M8, BD_MBI_S3_M1, BD_MBI_S3_M2, BD_MBI_S3_
    pd.concat(dataframes_Bd_S3[i], ignore_index=True) for i in range(24)
]

# Lista que contiene todos los DataFrames para brazo izquierdo
dataframes_brazo_izquierdo = [
    BI_MT_S1_M1, BI_MT_S1_M2, BI_MT_S1_M3, BI_MT_S1_M4, BI_MT_S1_M5, BI_MT_S1_M6, BI_MT_S1_M7, BI_MT_S1_M8,
    BI_MBI_S1_M1, BI_MBI_S1_M2, BI_MBI_S1_M3, BI_MBI_S1_M4, BI_MBI_S1_M5, BI_MBI_S1_M6, BI_MBI_S1_M7, BI_MBI_S1_M8,
    BI_MBS_S1_M1, BI_MBS_S1_M2, BI_MBS_S1_M3, BI_MBS_S1_M4, BI_MBS_S1_M5, BI_MBS_S1_M6, BI_MBS_S1_M7, BI_MBS_S1_M8,
    BI_MT_S2_M1, BI_MT_S2_M2, BI_MT_S2_M3, BI_MT_S2_M4, BI_MT_S2_M5, BI_MT_S2_M6, BI_MT_S2_M7, BI_MT_S2_M8,
    BI_MBI_S2_M1, BI_MBI_S2_M2, BI_MBI_S2_M3, BI_MBI_S2_M4, BI_MBI_S2_M5, BI_MBI_S2_M6, BI_MBI_S2_M7, BI_MBI_S2_M8,
    BI_MBS_S2_M1, BI_MBS_S2_M2, BI_MBS_S2_M3, BI_MBS_S2_M4, BI_MBS_S2_M5,  BI_MBS_S2_M6, BI_MBS_S2_M7, BI_MBS_S2_M8,
    BI_MT_S3_M1, BI_MT_S3_M2, BI_MT_S3_M3, BI_MT_S3_M4, BI_MT_S3_M5, BI_MT_S3_M6, BI_MT_S3_M7, BI_MT_S3_M8,
    BI_MBI_S3_M1, BI_MBI_S3_M2, BI_MBI_S3_M3, BI_MBI_S3_M4, BI_MBI_S3_M5, BI_MBI_S3_M6, BI_MBI_S3_M7, BI_MBI_S3_M8,
   BI_MBS_S3_M1, BI_MBS_S3_M2, BI_MBS_S3_M3, BI_MBS_S3_M4, BI_MBS_S3_M5, BI_MBS_S3_M6, BI_MBS_S3_M7, BI_MBS_S3_M8,
]

# Lista que contiene todos los DataFrames para brazo derecho
dataframes_brazo_derecho = [
    BD_MT_S1_M1, BD_MT_S1_M2, BD_MT_S1_M3, BD_MT_S1_M4, BD_MT_S1_M5, BD_MT_S1_M6, BD_MT_S1_M7, BD_MT_S1_M8,
    BD_MBI_S1_M1, BD_MBI_S1_M2, BD_MBI_S1_M3, BD_MBI_S1_M4, BD_MBI_S1_M5, BD_MBI_S1_M6, BD_MBI_S1_M7, BD_MBI_S1_M8,
    BD_MBS_S1_M1, BD_MBS_S1_M2, BD_MBS_S1_M3, BD_MBS_S1_M4, BD_MBS_S1_M5, BD_MBS_S1_M6, BD_MBS_S1_M7, BD_MBS_S1_M8,
    BD_MT_S2_M1, BD_MT_S2_M2, BD_MT_S2_M3, BD_MT_S2_M4, BD_MT_S2_M5, BD_MT_S2_M6, BD_MT_S2_M7, BD_MT_S2_M8,
    BD_MBI_S2_M1, BD_MBI_S2_M2, BD_MBI_S2_M3, BD_MBI_S2_M4, BD_MBI_S2_M5, BD_MBI_S2_M6, BD_MBI_S2_M7, BD_MBI_S2_M8,
    BD_MBS_S2_M1, BD_MBS_S2_M2, BD_MBS_S2_M3, BD_MBS_S2_M4, BD_MBS_S2_M5, BD_MBS_S2_M6, BD_MBS_S2_M7, BD_MBS_S2_M8,
    BD_MT_S3_M1, BD_MT_S3_M2, BD_MT_S3_M3, BD_MT_S3_M4, BD_MT_S3_M5, BD_MT_S3_M6, BD_MT_S3_M7, BD_MT_S3_M8,
    BD_MBI_S3_M1, BD_MBI_S3_M2, BD_MBI_S3_M3, BD_MBI_S3_M4, BD_MBI_S3_M5, BD_MBI_S3_M6, BD_MBI_S3_M7, BD_MBI_S3_M8,
    BD_MBS_S3_M1, BD_MBS_S3_M2, BD_MBS_S3_M3, BD_MBS_S3_M4, BD_MBS_S3_M5, BD_MBS_S3_M6, BD_MBS_S3_M7, BD_MBS_S3_M8,
]
nombres_brazo_izquierdo = [nombre for nombre, valor in globals().items() if isinstance(valor, pd.DataFrame) and any(valor.equals(df) for df i
nombres_brazo_derecho = [nombre for nombre, valor in globals().items() if isinstance(valor, pd.DataFrame) and any(valor.equals(df) for df in
```

Gráficas de las señales por cada dataframe

```
def plot_dataframes(dataframes, nombres):
    num_plots = len(dataframes)
    num_rows = math.ceil(num_plots / 3)
    fig, axes = plt.subplots(num_rows, 3, figsize=(18, num_rows * 3))

    for i, (variable, nombre) in enumerate(zip(dataframes, nombres)):
        row = i // 3
        col = i % 3
        axes[row, col].plot(variable.T)
        axes[row, col].set_title(nombre)
        axes[row, col].set_xlabel("Datos en el tiempo")
        axes[row, col].set_ylabel("Valor")

    plt.tight_layout()
    plt.show()

# Plot brazo izquierdo
plot_dataframes(dataframes_brazo_izquierdo, nombres_brazo_izquierdo)

# Plot brazo derecho
plot_dataframes(dataframes_brazo_derecho, nombres_brazo_derecho)
```

CNN Categórica

```
# Combinaciones
BI_MT_dataframes = [
    ['BI_MT_S1_M1', 'BI_MT_S2_M1', 'BI_MT_S3_M1'],
    ['BI_MT_S1_M2', 'BI_MT_S2_M2', 'BI_MT_S3_M2'],
    ['BI_MT_S1_M3', 'BI_MT_S2_M3', 'BI_MT_S3_M3'],
    ['BI_MT_S1_M4', 'BI_MT_S2_M4', 'BI_MT_S3_M4'],
    ['BI_MT_S1_M5', 'BI_MT_S2_M5', 'BI_MT_S3_M5'],
    ['BI_MT_S1_M6', 'BI_MT_S2_M6', 'BI_MT_S3_M6'],
    ['BI_MT_S1_M7', 'BI_MT_S2_M7', 'BI_MT_S3_M7'],
    ['BI_MT_S1_M8', 'BI_MT_S2_M8', 'BI_MT_S3_M8']
]
BI_MBI_dataframes = [
    ['BI_MBI_S1_M1', 'BI_MBI_S2_M1', 'BI_MBI_S3_M1'],
    ['BI_MBI_S1_M2', 'BI_MBI_S2_M2', 'BI_MBI_S3_M2'],
    ['BI_MBI_S1_M3', 'BI_MBI_S2_M3', 'BI_MBI_S3_M3'],
    ['BI_MBI_S1_M4', 'BI_MBI_S2_M4', 'BI_MBI_S3_M4'],
    ['BI_MBI_S1_M5', 'BI_MBI_S2_M5', 'BI_MBI_S3_M5'],
    ['BI_MBI_S1_M6', 'BI_MBI_S2_M6', 'BI_MBI_S3_M6'],
    ['BI_MBI_S1_M7', 'BI_MBI_S2_M7', 'BI_MBI_S3_M7'],
    ['BI_MBI_S1_M8', 'BI_MBI_S2_M8', 'BI_MBI_S3_M8']
]

BI_MBS_dataframes = [
    ['BI_MBS_S1_M1', 'BI_MBS_S2_M1', 'BI_MBS_S3_M1'],
    ['BI_MBS_S1_M2', 'BI_MBS_S2_M2', 'BI_MBS_S3_M2'],
    ['BI_MBS_S1_M3', 'BI_MBS_S2_M3', 'BI_MBS_S3_M3'],
    ['BI_MBS_S1_M4', 'BI_MBS_S2_M4', 'BI_MBS_S3_M4'],
    ['BI_MBS_S1_M5', 'BI_MBS_S2_M5', 'BI_MBS_S3_M5'],
    ['BI_MBS_S1_M6', 'BI_MBS_S2_M6', 'BI_MBS_S3_M6'],
    ['BI_MBS_S1_M7', 'BI_MBS_S2_M7', 'BI_MBS_S3_M7'],
    ['BI_MBS_S1_M8', 'BI_MBS_S2_M8', 'BI_MBS_S3_M8']
]
BD_MT_dataframes = [
    ['BD_MT_S1_M1', 'BD_MT_S2_M1', 'BD_MT_S3_M1'],
    ['BD_MT_S1_M2', 'BD_MT_S2_M2', 'BD_MT_S3_M2'],
    ['BD_MT_S1_M3', 'BD_MT_S2_M3', 'BD_MT_S3_M3'],
    ['BD_MT_S1_M4', 'BD_MT_S2_M4', 'BD_MT_S3_M4'],
    ['BD_MT_S1_M5', 'BD_MT_S2_M5', 'BD_MT_S3_M5'],
    ['BD_MT_S1_M6', 'BD_MT_S2_M6', 'BD_MT_S3_M6'],
    ['BD_MT_S1_M7', 'BD_MT_S2_M7', 'BD_MT_S3_M7'],
    ['BD_MT_S1_M8', 'BD_MT_S2_M8', 'BD_MT_S3_M8']
]
BD_MBI_dataframes = [
    ['BD_MBI_S1_M1', 'BD_MBI_S2_M1', 'BD_MBI_S3_M1'],
    ['BD_MBI_S1_M2', 'BD_MBI_S2_M2', 'BD_MBI_S3_M2'],
    ['BD_MBI_S1_M3', 'BD_MBI_S2_M3', 'BD_MBI_S3_M3'],
    ['BD_MBI_S1_M4', 'BD_MBI_S2_M4', 'BD_MBI_S3_M4'],
    ['BD_MBI_S1_M5', 'BD_MBI_S2_M5', 'BD_MBI_S3_M5'],
    ['BD_MBI_S1_M6', 'BD_MBI_S2_M6', 'BD_MBI_S3_M6'],
    ['BD_MBI_S1_M7', 'BD_MBI_S2_M7', 'BD_MBI_S3_M7'],
    ['BD_MBI_S1_M8', 'BD_MBI_S2_M8', 'BD_MBI_S3_M8']
]

BD_MBS_dataframes = [
    ['BD_MBS_S1_M1', 'BD_MBS_S2_M1', 'BD_MBS_S3_M1'],
    ['BD_MBS_S1_M2', 'BD_MBS_S2_M2', 'BD_MBS_S3_M2'],
    ['BD_MBS_S1_M3', 'BD_MBS_S2_M3', 'BD_MBS_S3_M3'],
    ['BD_MBS_S1_M4', 'BD_MBS_S2_M4', 'BD_MBS_S3_M4'],
    ['BD_MBS_S1_M5', 'BD_MBS_S2_M5', 'BD_MBS_S3_M5'],
    ['BD_MBS_S1_M6', 'BD_MBS_S2_M6', 'BD_MBS_S3_M6'],
    ['BD_MBS_S1_M7', 'BD_MBS_S2_M7', 'BD_MBS_S3_M7'],
    ['BD_MBS_S1_M8', 'BD_MBS_S2_M8', 'BD_MBS_S3_M8']
]

# Crea un arreglo tridimensional para los bloques de datos
bloques_datos_G = []

# Contador para etiquetar los bloques del 1 al 8 para cada músculo
label_counter = 1

# Itera sobre las combinaciones y sus DataFrames correspondientes
for combinacion_dfs in BI_MT_dataframes + BI_MBI_dataframes + BI_MBS_dataframes + BD_MT_dataframes + BD_MBI_dataframes + BD_MBS_dataframes :
    # Obtener los DataFrames correspondientes a la combinación actual
    dfs = [globals()[df_name] for df_name in combinacion_dfs]

    # Verificar que los DataFrames tengan la misma longitud
    if not all(len(df) == len(dfs[0]) for df in dfs):

        print("Los DataFrames de la combinación no tienen la misma longitud.")
        continue

     # Crea un arreglo tridimensional para el bloque de datos actual
     bloque_datos_G = np.empty((len(dfs[0]), 40, 3), dtype=object)

     # Itera sobre las filas de los DataFrames
     for i in range(len(dfs[0])):
         # Iterar sobre los 40 datos en cada señal
         for j in range(40):
             # Asignar los valores de los DataFrames al bloque de datos, manejando valores None o NaN
             for k in range(3):
                 value = dfs[k].iloc[i, j]
                 if pd.isnull(value):
                     value = None
                 bloque_datos_G[i, j, k] = value

     # Agregar el bloque de datos al arreglo de bloques con etiqueta
     bloques_datos_G.append({
         "label": label_counter,
         "data": bloque_datos_G
     })

     # Incrementar el contador de etiquetas y reiniciar si llega a 8
     label_counter += 1
     if label_counter > 8:
         label_counter = 1
```

```
# Verifica la forma de todos los bloques de datos
for i, bloque in enumerate(bloques_datos_G, start=1):
    print(f"Forma del bloque de datos {i}:", bloque["data"].shape)

# Crea un vector de etiquetas del 1 al número de bloques para asociar cada bloque con su etiqueta
etiquetas = np.array([bloque["label"] for bloque in bloques_datos_G])

# Verifica la forma del arreglo de bloques de datos
print("Forma del arreglo de bloques de datos:", bloques_datos_G[0]["data"].shape)

# Imprime el tamaño del dataset que alberga los bloques creados
print("Tamaño del dataset:", len(bloques_datos_G))


# Crear un arreglo tridimensional para el cubo final
cubo_final_G = np.empty((2880, 40, 3), dtype=object)

# Índice para rastrear la posición actual en el cubo final
indice = 0

# Itera sobre los bloques existentes
for bloque in bloques_datos_G:
    etiqueta = bloque['label']
    datos_bloque_G = bloque['data']

    # Añadir los datos del bloque al cubo final en la posición correcta
    cubo_final_G[indice:indice+len(datos_bloque_G), :, :] = datos_bloque_G

    # Incrementa el índice para la próxima etiqueta
    indice += len(datos_bloque_G)


print("Forma del cubo final:", cubo_final_G.shape)
print("Tamaño del cubo final:", indice)


etiquetas_cubo_final_G = np.repeat(np.arange(1, 9), 360)

# Convertir las etiquetas a one-hot encoding
y = to_categorical(etiquetas_cubo_final_G - 1, num_classes=8)

# Convertir el cubo final a un formato adecuado para la red convolucional
X = cubo_final_G.reshape(-1, 40, 3)

trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.2, random_state=42)
n_timesteps, n_features, n_outputs = trainX.shape[1], trainX.shape[2], trainy.shape[1]
print(n_timesteps, n_features, n_outputs)

# Modelo Red Convolucional
epochs, batch_size = 35, 10
model2 = Sequential()
model2.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(n_timesteps,n_features)))
model2.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
model2.add(Dropout(0.5))
model2.add(MaxPooling1D(pool_size=2))
model2.add(Flatten())
model2.add(Dense(100, activation='relu'))
model2.add(Dense(n_outputs, activation='softmax'))
model2.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model2.summary()

trainX = trainX.astype('float32')
trainy = trainy.astype('float32')
testX = testX.astype('float32')
testy = testy.astype('float32')

# Entrenar el modelo
model2.fit(trainX, trainy, epochs=epochs, batch_size=batch_size, validation_data=(testX, testy))
```

Reporte y Comprobación del Modelo

```
# Obtener las predicciones del conjunto de pruebas
predicciones = model2.predict(testX)
predicciones_indices = np.argmax(predicciones, axis=1)
etiquetas_indices = np.argmax(testy, axis=1)

# Crear un diccionario para almacenar las predicciones por categoría
predicciones_por_categoria = {i: [] for i in range(1, 9)}

# Llenar el diccionario con las predicciones correctas para cada categoría
for i in range(len(predicciones_indices)):
    categoria_predicha = predicciones_indices[i] + 1
    categoria_real = etiquetas_indices[i] + 1
    if categoria_predicha == categoria_real:
        predicciones_por_categoria[categoria_real].append(True)
    else:
        predicciones_por_categoria[categoria_real].append(False)

# Calcular la precisión para cada categoría
precisiones_por_categoria = {i: sum(predicciones_por_categoria[i]) / len(predicciones_por_categoria[i]) for i in range(1, 9)}

# Imprimir las precisiones por categoría
for categoria, precision in precisiones_por_categoria.items():
    print(f"Categoría {categoria}: Precisión = {round(precision, 3)}")

# Seleccionar un conjunto específico de señales del conjunto de pruebas (por ejemplo, las primeras 10 señales)
señales_a_verificar = testX[:30]
etiquetas_reales = testy[:30]


# Obtener las predicciones para las señales seleccionadas
predicciones = model2.predict(señales_a_verificar)
predicciones_indices = np.argmax(predicciones, axis=1)

# Imprimir las precisiones por categoría
for i in range(len(predicciones_indices)):
    categoria_predicha = predicciones_indices[i] + 1
    categoria_real = np.argmax(etiquetas_reales[i]) + 1
    if categoria_predicha == categoria_real:
        print(f"Señal {i+1} (Categoría {categoria_real}): Predicción correcta. Categoría predicha: {categoria_predicha}")
    else:
        print(f"Señal {i+1} (Categoría {categoria_real}): Predicción incorrecta. Categoría predicha: {categoria_predicha}")

import pandas as pd

# Crear un DataFrame con las precisiones por categoría
precisiones_df = pd.DataFrame(precisiones_por_categoria.items(), columns=['Categoría', 'Precisión'])
precisiones_df['Precisión'] = precisiones_df['Precisión'].map(lambda x: round(x, 3))

# Imprimir el DataFrame como una tabla
print("Precisiones por Categoría:")
print(precisiones_df.to_string(index=False))

# Crear un DataFrame con las señales verificadas y sus predicciones
resultados_df = pd.DataFrame(columns=['Señal', 'Categoría Real', 'Predicción', 'Correcto'])
for i in range(len(predicciones_indices)):
    categoria_predicha = predicciones_indices[i] + 1
    categoria_real = np.argmax(etiquetas_reales[i]) + 1
    correcto = categoria_predicha == categoria_real
    resultados_df = resultados_df.append({'Señal': i+1, 'Categoría Real': categoria_real, 'Predicción': categoria_predicha, 'Correcto': corr

# Imprimir el DataFrame como una tabla
print("\nResultados de las Señales Verificadas:")
print(resultados_df.to_string(index=False))
```

CNN Predicción Temporal

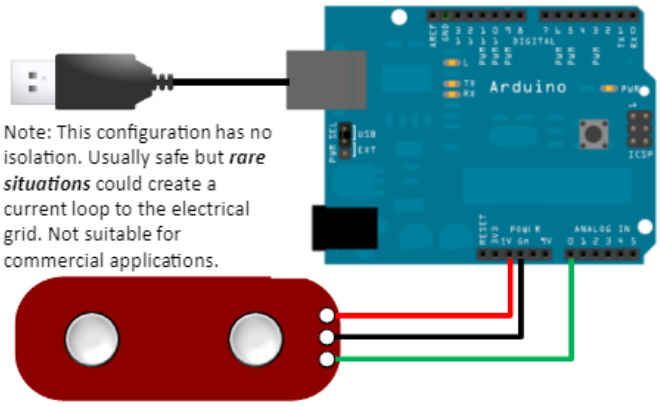
```
models_brazo_izquierdo = []
models_brazo_derecho = []

for df in dataframes_brazo_izquierdo:
    X = df.values
    X = X.reshape(X.shape[0], X.shape[1], 1)

    y_train = X.copy()
    y_train[:, :25] = 0

    y_test = X.copy()
    y_test[:, :25] = 0

    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=5, activation='relu', input_shape=(40, 1)))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Conv1D(filters=128, kernel_size=5, activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Conv1D(filters=256, kernel_size=5, activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(40))
    model.compile(optimizer='adam', loss='mse')

    model.fit(X, y_train, epochs=70, batch_size=32, verbose=0)

    models_brazo_izquierdo.append(model)


for df in dataframes_brazo_derecho:
    X = df.values
    X = X.reshape(X.shape[0], X.shape[1], 1)

    y_train = X.copy()
    y_train[:, :25] = 0

    y_test = X.copy()
    y_test[:, :25] = 0

    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=5, activation='relu', input_shape=(40, 1)))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Conv1D(filters=128, kernel_size=5, activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Conv1D(filters=256, kernel_size=5, activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(40))
    model.compile(optimizer='adam', loss='mse')

    model.fit(X, y_train, epochs=70, batch_size=32, verbose=0)

    models_brazo_derecho.append(model)
```

Comprobación del modelo

```
def plot_predictions_and_models(dataframes_brazo_izquierdo, dataframes_brazo_derecho, models_brazo_izquierdo, models_brazo_derecho, names_iz
    # Graficar para el brazo izquierdo
    for df, model, name in zip(dataframes_brazo_izquierdo, models_brazo_izquierdo, names_izquierdo):
        X = df.values
        X = X.reshape(X.shape[0], X.shape[1], 1)

        y_pred = model.predict(X)

        plt.figure(figsize=(12, 6))
        plt.plot(range(40), X[0].flatten(), label='Original')
        plt.plot(range(25, 40), np.zeros(15), label='Faltante')
        plt.plot(range(25, 40), y_pred[0, 25:].flatten(), label='Predicción')
        plt.legend()
        plt.title(f'Predicción de Señal - Brazo Izquierdo - {name}')
        plt.show()

    # Graficar para el brazo derecho
    for df, model, name in zip(dataframes_brazo_derecho, models_brazo_derecho, names_derecho):
        X = df.values
        X = X.reshape(X.shape[0], X.shape[1], 1)

        y_pred = model.predict(X)

        plt.figure(figsize=(12, 6))
        plt.plot(range(40), X[0].flatten(), label='Original')
        plt.plot(range(25, 40), np.zeros(15), label='Faltante')
        plt.plot(range(25, 40), y_pred[0, 25:].flatten(), label='Predicción')
        plt.legend()
        plt.title(f'Predicción de Señal - Brazo Derecho - {name}')
        plt.show()

# Nombres de los DataFrames
nombres_brazo_izquierdo = [nombre for nombre, valor in globals().items() if isinstance(valor, pd.DataFrame) and any(valor.equals(df) for df
nombres_brazo_derecho = [nombre for nombre, valor in globals().items() if isinstance(valor, pd.DataFrame) and any(valor.equals(df) for df in
plot_predictions_and_models(dataframes_brazo_izquierdo, dataframes_brazo_derecho, models_brazo_izquierdo, models_brazo_derecho, nombres_braz
```

ANEXO D: HOJA DE DATOS - MYOWARE

d) Grid powered. ***Warning: No isolation.***

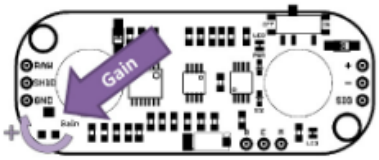


Note: This configuration has no isolation. Usually safe but ***rare situations*** could create a current loop to the electrical grid. Not suitable for commercial applications.

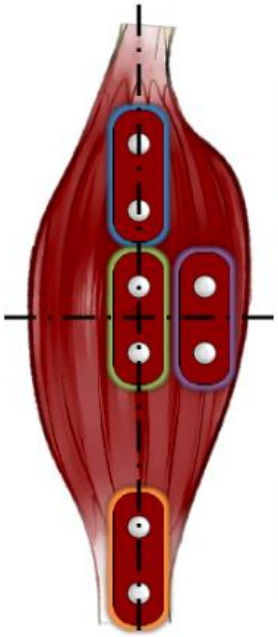
Adjusting the gain

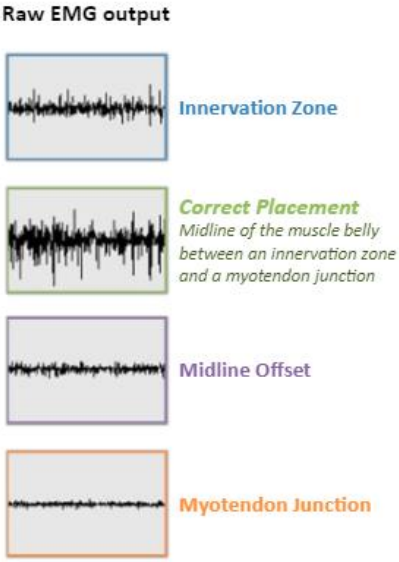
We recommend for users to get their sensor setup working reliably prior to adjusting the gain. The default gain setting should be appropriate for most applications.
To adjust the gain, locate the gain potentiometer in the lower left corner of the sensor (marked as "GAIN"). Using a Phillips screwdriver, turn the potentiometer counterclockwise to increase the output gain; turn the potentiometer clockwise to reduce the gain.

Note: In order to reduce the required voltage for the sensor, the redesign switch out a JFET amplifier for a CMOS amplifier. However CMOS amplifiers tend to have slower recovery times when saturated. Therefore, we advise users to adjust the gain such that the output signal will not saturate the amplifier.



Why is electrode placement important?



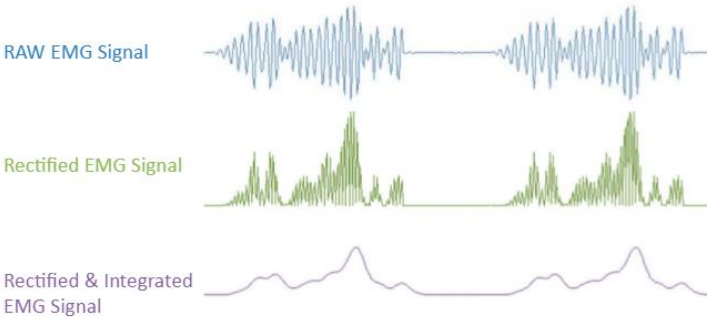


Electrical Specifications

Parameter	Min	TYP	Max
Supply Voltage	+2.9V	+3.3V or +5V	+5.7V
Adjustable Gain Potentiometer	0.01 Ω	50 kΩ	100 kΩ
Output Signal Voltage EMG Envelope Raw EMG (centered about +Vs/2)	 0V 0V	 -- --	 +Vs +Vs
Input Impedance	--	110 GΩ	--
Supply Current	--	9 mA	14 mA
Common Mode Rejection Ratio (CMRR)	--	110	--
Input Bias	--	1 pA	--

RAW EMG vs EMG Envelope

Our Muscle Sensors are designed to be used directly with a microcontroller. Therefore, our sensors primary output is not a RAW EMG signal but rather an amplified, rectified, and integrated signal (AKA the EMG's envelope) that will work well with a microcontroller's analog-to-digital converter (ADC). This difference is illustrated below using a representative EMG signal. *Note: Actual sensor output not shown.*



ANEXO E: HOJA DE DATOS - TARJETA DE DESARROLLO ESP32S

Table 1 provides the specifications of ESP-32S.

Table 1 ESP-32S Specifications

Categories	Items	Values
WiFi	Standards	
	Protocles	802.11 b/g/n/d/e/i/k/r (802.11n up to 150 Mbps)
	Frequency Range	2.4GHz-2.5GHz (2400M-2483.5M)
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with -98 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC
Hardware	Module interface	SD card, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, I2C, IR
		GPIO, capacitive touch sensor, ADC, DAC, LNA pre-amplier
	On-chip sensor	3.0~3.6V
	On-board clock	Average value: 80mA
	Operating voltage	-40°~125°
	Operating current	Normal temperature
	Operating temperature range	14.3mm*24.8mm*3mm
	Ambient temperature range	N/A
	Package size	
Software	Wi-Fi mode	Station/softAP/SoftAP+station/P2P
	Security	WPA/WPA2/WPA2-Enterprise/WPS
	Encryption	AES/RSA/ECC/SHA
	Firmware Upgrade	UART Download / OTA (via network) / download and write firmware via host

	Ssoftware Development	Supports Cloud Server Development / SDK for custom firmware development
	Network Protocols	IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT
	User Configuration	AT instruction set, cloud server, Android/iOS App

Note:

* ESP-32S with high temperature range option (-40°C ~ 125°C) is available for custom order.

2. Pin Definitions

2.1 Pin Layout

Figure 1: Top and Side View of ESP-32S

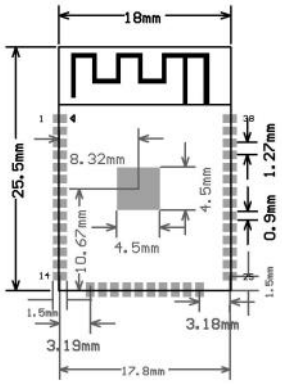


Table 2: ESP-32S Dimensions

Length	Width	Height	PAD Size(Bottom)	Pin Pitch	Shielding can height	PCB thickness
18mm	25.5mm	2.8 ± 0.1 mm	0.45 mm x 0.9 mm	1.27mm	2 mm	0.8 ± 0.1 mm

2.2 Pin Description

ESP-32S has 38 pins. See pin definitions in Table 3.

Table 3 Pin Descriptions

NO	Pin Name	Function
1	GND	Ground
2	3V3	Power supply
3	EN	Chip-enable signal. Active high
4	SENSOR_VP	GPI36, SENSOR_VP, ADC_H, ADC1_CH0, RTC_GPIO0
5	SENSOR_VN	GPI39, SENSOR_VN, ADC1_CH3, ADC_H, RTC_GPIO3
6	IO34	GPI34, ADC1_CH6, RTC_GPIO4
7	IO35	GPI35, ADC1_CH7, RTC_GPIO5
8	IO32	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
9	IO33	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
10	IO25	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
11	IO26	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1

12	IO27	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
13	IO14	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
14	IO12	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
15	GND	Ground
16	IO13	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
17	SHD/SD2	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
18	SHD/SD3	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
19	SCS/CMD	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
20	SCK/CLK	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
21	SDO/SD0	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
22	SDI/SD1	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
23	IO15	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
24	IO2	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
25	IO0	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
26	IO4	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1,

		SD_DATA1, EMAC_TX_ER
27	IO16	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
28	IO17	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
29	IO5	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
30	IO18	GPIO18, VSPICLK, HS1_DATA7
31	IO19	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
32	NC	-
33	IO21	GPIO21, VSPIHD, EMAC_TX_EN
34	RXD0	GPIO3, U0RXD, CLK_OUT2
35	TXDO	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
36	IO22	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
37	IO23	GPIO23, VSPID, HS1_STROBE
38	GND	Ground

Table 4: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)					
Pin	Default	3.3V		1.8V	
MTDI	Pull-down	0		1	
Booting Mode					
Pin	Default	SPI Flash Boot		Download Boot	
GPIO0	Pull-up	1		0	
GPIO2	Pull-down	Don't-care		0	
Debugging Log on U0TXD During Booting					
Pin	Default	U0TXD Toggling		U0TXD Silent	
MTDO	Pull-up	1		0	
Timing of SDIO Slave					
Pin	Default	Falling-edge Input Falling-edge Output	Falling-edge Input Rising-edge Output	Rising-edge Input Falling-edge Output	Rising-edge Input Rising-edge Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

Note:
Firmware can configure register bits to change the settings of "Voltage of Internal LDO (VDD_SDIO)" and "Timing of SDIO
Slave" after booting

Table 5: Power Consumption by Power Modes

Power mode	Comment	Power consumption
Active mode (RF working)	Wi-Fi Tx packet 13 dBm ~ 21 dBm	160 ~ 260 mA
	Wi-Fi / BT Tx packet 0 dBm	120 mA
	Wi-Fi / BT Rx and listening	80 ~ 90 mA
	Association sleep pattern (by Lightsleep)	0.9 mA@DTIM3, 1.2 mA@DTIM1
Modem-sleep mode	The CPU is powered on.	Max speed: 20 mA
		Normal: 5 ~ 10 mA
		Slow speed: 3 mA
Light-sleep mode	-	0.8 mA
Deep-sleep mode	The ULP-coprocessor is powered on	0.15 mA
	ULP sensor-monitored pattern	25 μA @1% duty
	RTC timer + RTC memories	20 μA
Hibernation mode	RTC timer only	2.5 μA

ANEXO F: COMPARACIÓN DE RESPUESTA ANTE DIFERENTES MOVIMIENTOS ESPECÍFICOS

Movimiento	Sensor 1	Sensor 2	Sensor 3
Abrir mano	1,72V	0,95V	2,6V
	1,73V	0,92V	3,55V
	1,79V	1,0V	3,6V
	1,85V	1,02V	3,63V
	2,3V	1,19V	3,87V
	2,35V	1,24V	3,92V
	2,39V	1,32V	3,98V
	2,43V	1,47V	4,05V
	2,46V	1,56V	4,17V
	2,51V	1,63V	4,22V
Cerrar mano	4,53V	2,49V	4,67V
	3,67V	2,35V	4,58V
	3,41V	2,12V	4,47V
	3,29V	1,94V	4,32V
	3,09V	1,86V	4,28V
	2,72V	1,77V	4,21V
	2,65V	1,64V	4,14V
	2,56V	1,59V	3,98V
	2,48V	1,52V	3,89V
	2,29V	1,47V	3,74V
Levantar brazo	3,41V	1,4V	3,61V
	3,36V	1,34V	3,57V
	2,94V	1,28V	3,49V
	2,86V	1,16V	3,43V
	2,67V	1,09V	3,25V
	2,39V	1,04V	3,18V
	2,17V	1,01V	2,84V
	1,48V	0,98V	2,67V
	1,23V	0,94V	2,47V
	1,05V	0,9V	2,28V
Bajar brazo	1,31V	0,79V	1,73V
	1,32V	0,74V	1,68V
	1,36V	0,71V	1,62V
	1,23V	0,69V	1,57V
	1,03V	0,67V	1,55V

	0,98V	0,59V	1,49V
	0,92V	0,48V	1,42V
	0,89V	0,37V	1,38V
	0,82V	0,28V	1,31V
	0,76V	0,19V	1,26V
Agarrar objeto	2,76V	1,06V	3,53V
	2,43V	1,01V	3,28V
	2,0V	0,94V	3,12V
	1,48V	0,76V	2,89V
	1,35V	0,64V	2,72V
	1,29V	0,56V	2,62V
	1,21V	0,48V	2,56V
	1,17V	0,37V	2,49V
	1,10V	0,25V	2,45V
	1,05V	0,20V	2,34V
Levantar objeto	4,24V	2,32V	4,43V
	3,59V	2,01V	4,38V
	3,38V	1,82V	4,31V
	2,17V	1,78V	4,29V
	1,91V	1,64V	3,93V
	1,87V	1,56V	3,79V
	1,76V	1,47V	3,63V
	1,61V	1,32V	3,56V
	1,54V	1,26V	3,42V
	1,45V	1,19V	3,29V
Bajar objeto	1,66V	1,17V	2,24V
	1,43V	1,04V	2,21V
	1,29V	0,77V	2,14V
	1,24V	0,68V	1,94V
	1,19V	0,56V	1,8V
	1,07V	0,43V	1,76V
	1,02V	0,38V	1,67V
	0,97V	0,26V	1,58V
	0,82V	0,19V	1,43V
	0,74V	0,12V	1,39V
Soltar objeto	0,96V	0,53V	1,72V
	0,86V	0,49V	1,68V
	0,84V	0,46V	1,63V
	0,73V	0,44V	1,57V

0,65V	0,39V	1,42V
0,58V	0,31V	1,34V
0,47V	0,27V	1,25V
0,42V	0,22V	1,18V
0,36V	0,17V	1,11V
0,29V	0,10V	0,98V

Realizado por: Vaca-Paguay, 2024.

ANEXO G: ANÁLISIS DE CORRELACIÓN COMBINACIÓN POR ZONA.

Análisis de correlación entre sensores (brazo derecho)

N°	Combinaciones	Coeficiente de correlación
1	'BD_MT_S1_M1', 'BD_MT_S2_M1', 'BD_MT_S3_M1'	0.675
2	'BD_MT_S1_M2', 'BD_MT_S2_M2', 'BD_MT_S3_M2'	0.739
3	'BD_MT_S1_M3', 'BD_MT_S2_M3', 'BD_MT_S3_M3'	0.649
4	'BD_MT_S1_M4', 'BD_MT_S2_M4', 'BD_MT_S3_M4'	0.614
5	'BD_MT_S1_M5', 'BD_MT_S2_M5', 'BD_MT_S3_M5'	0.721
6	'BD_MT_S1_M6', 'BD_MT_S2_M6', 'BD_MT_S3_M6'	0.549
7	'BD_MT_S1_M7', 'BD_MT_S2_M7', 'BD_MT_S3_M7'	0.734
8	'BD_MT_S1_M8', 'BD_MT_S2_M8', 'BD_MT_S3_M8'	0.806
9	'BD_MBI_S1_M1', 'BD_MBI_S2_M1', 'BD_MBI_S3_M1'	0.722
10	'BD_MBI_S1_M2', 'BD_MBI_S2_M2', 'BD_MBI_S3_M2'	0.789
11	'BD_MBI_S1_M3', 'BD_MBI_S2_M3', 'BD_MBI_S3_M3'	0.707
12	'BD_MBI_S1_M4', 'BD_MBI_S2_M4', 'BD_MBI_S3_M4'	0.766
13	'BD_MBI_S1_M5', 'BD_MBI_S2_M5', 'BD_MBI_S3_M5'	0.762
14	'BD_MBI_S1_M6', 'BD_MBI_S2_M6', 'BD_MBI_S3_M6'	0.580
15	'BD_MBI_S1_M7', 'BD_MBI_S2_M7', 'BD_MBI_S3_M7'	0.562
16	'BD_MBI_S1_M8', 'BD_MBI_S2_M8', 'BD_MBI_S3_M8'	0.343
17	'BD_MBS_S1_M1', 'BD_MBS _S2_M1','BD_MBS_S3_M1'	0.715
18	'BD_MBS_S1_M2', 'BD_ MBS_S2_M2', BD_MBS_S3_M2'	0.487
19	'BD_MBS _S1_M3', 'BD_MBS_S2_M3', BD_MBS_S3_M3'	0.590
20	'BD_MBS_S1_M4', 'BD_MBS_S2_M4', 'BD_MBS_S3_M4'	0.612
21	'BD_MBI_S1_M5', 'BD_MBS_S2_M5', 'BD_MBS_S3_M5'	0.464
22	'BD_MBI_S1_M6', 'BD_MBS_S2_M6', 'BD_MBS_S3_M6'	0.631
23	'BD_MBI_S1_M7', 'BD_MBS_S2_M7', 'BD_MBS_S3_M7'	0.329
24	'BD_MBI_S1_M8', 'BD_MBS_S2_M8', 'BD_MBS_S3_M8'	0.500

Realizado por: Vaca-Paguay, 2024.

Análisis de correlación entre sensores (brazo izquierdo)

N°	Combinaciones	Coeficiente de correlación
1	'BI_MT_S1_M1', 'BI_MT_S2_M1', 'BI_MT_S3_M1'	0.669
2	'BI_MT_S1_M2', 'BI_MT_S2_M2', 'BI_MT_S3_M2'	0.711
3	'BI_MT_S1_M3', 'BI_MT_S2_M3', 'BI_MT_S3_M3'	0.716
4	'BI_MT_S1_M4', 'BI_MT_S2_M4', 'BI_MT_S3_M4'	0.831
5	'BI_MT_S1_M5', 'BI_MT_S2_M5', 'BI_MT_S3_M5'	0.757
6	'BI_MT_S1_M6', 'BI_MT_S2_M6', 'BI_MT_S3_M6'	0.728
7	'BI_MT_S1_M7', 'BI_MT_S2_M7', 'BI_MT_S3_M7'	0.785
8	'BI_MT_S1_M8', 'BI_MT_S2_M8', 'BI_MT_S3_M8'	0.786
9	'BI_MBI_S1_M1', 'BI_MBI_S2_M1', 'BI_MBI_S3_M1'	0.742
10	'BI_MBI_S1_M2', 'BI_MBI_S2_M2', 'BI_MBI_S3_M2'	0.765
11	'BI_MBI_S1_M3', 'BI_MBI_S2_M3', 'BI_MBI_S3_M3'	0.430

12	'BI_MBI_S1_M4', 'BI_MBI_S2_M4', 'BI_MBI_S3_M4'	0.468
13	'BI_MBI_S1_M5', 'BI_MBI_S2_M5', 'BI_MBI_S3_M5'	0.517
14	'BI_MBI_S1_M6', 'BI_MBI_S2_M6', 'BI_MBI_S3_M6'	0.549
15	'BI_MBI_S1_M7', 'BI_MBI_S2_M7', 'BI_MBI_S3_M7'	0.582
16	'BI_MBI_S1_M8', 'BI_MBI_S2_M8', 'BI_MBI_S3_M8'	0.712
17	'BI_MBS_S1_M1', 'BI_MBS_S2_M1','BI _MBS_S3_M1'	0.602
18	'BI_MBS_S1_M2', 'BI_MBS_S2_M2', BI_MBS_S3_M2'	0.532
19	'BI_MBS_S1_M3', 'BI_MBS_S2_M3', BI_MBS_S3_M3'	0.527
20	'BI_MBS_S1_M4', 'BI_MBS_S2_M4', 'BI_MBS_S3_M4'	0.702
21	'BI_MBI_S1_M5', 'BI_MBS_S2_M5', 'BI_MBS_S3_M5'	0.580
22	'BI_MBI_S1_M6', 'BI_MBS_S2_M6', 'BI_MBS_S3_M6'	0.661
23	'BI_MBI_S1_M7', 'BI_MBS_S2_M7', 'BI_MBS_S3_M7'	0.520
24	'BI_MBI_S1_M8', 'BI_MBS_S2_M8', 'BI_MBS_S3_M8'	0.521

Realizado por: Vaca-Paguay, 2024.

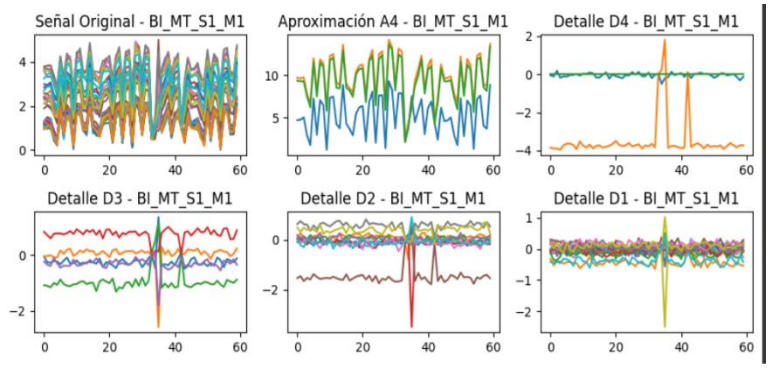
ANEXO H: ANÁLISIS DE CORRELACIÓN (PRUEBA ESPEJO)

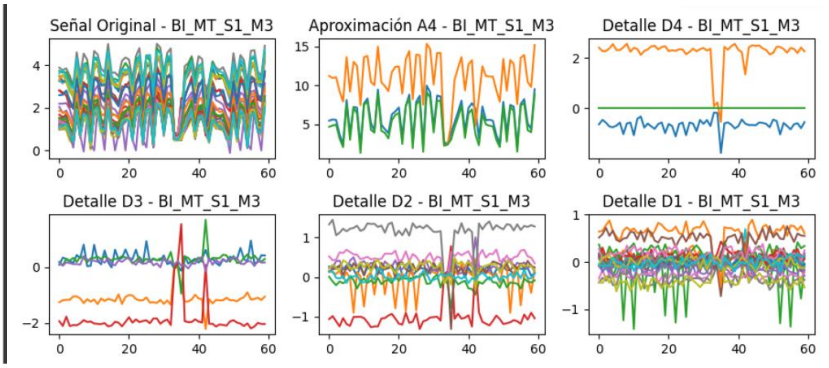
N°	Combinaciones	Coeficiente de correlación
1	BI_MT_S1_M1 y BD_MT_S1_M1	-0.067
2	BI_MT_S1_M2 y BD_MT_S1_M2	0.107
3	BI_MT_S1_M3 y BD_MT_S1_M3	0.028
4	BI_MT_S1_M4 y BD_MT_S1_M4	-0.030
5	BI_MT_S1_M5 y BD_MT_S1_M5	0.173
6	BI_MT_S1_M6 y BD_MT_S1_M6	-0.195
7	BI_MT_S1_M7 y BD_MT_S1_M7	0.292
8	BI_MT_S1_M8 y BD_MT_S1_M8	0.091
9	BI_MBI_S1_M1 y BD_MBI_S1_M1	0.003
10	BI_MBI_S1_M2 y BD_MBI_S1_M2	0.141
11	BI_MBI_S1_M3 y BD_MBI_S1_M3	0.211
12	BI_MBI_S1_M4 y BD_MBI_S1_M4	0.289
13	BI_MBI_S1_M5 y BD_MBI_S1_M5	0.155
14	BI_MBI_S1_M6 y BD_MBI_S1_M6	-0.109
15	BI_MBI_S1_M7 y BD_MBI_S1_M7	0.375
16	BI_MBI_S1_M8 y BD_MBI_S1_M8	-0.315
17	BI_MBS_S1_M1 y BD_MBS_S1_M1	0.058
18	BI_MBS_S1_M2 y BD_MBS_S1_M2	-0.179
19	BI_MBS_S1_M3 y BD_MBS_S1_M3	0.103
20	BI_MBS_S1_M4 y BD_MBS_S1_M4	-0.114
21	BI_MBS_S1_M5 y BD_MBS_S1_M5	0.090
22	BI_MBS_S1_M6 y BD_MBS_S1_M6	0.048
23	BI_MBS_S1_M7 y BD_MBS_S1_M7	-0.297
24	BI_MBS_S1_M8 y BD_MBS_S1_M8	0.017
25	BI_MT_S2_M1 y BD_MT_S2_M1	0.087
26	BI_MT_S2_M2 y BD_MT_S2_M2	0.134
27	BI_MT_S2_M3 y BD_MT_S2_M3	0.275
28	BI_MT_S2_M4 y BD_MT_S2_M4	0.153
29	BI_MT_S2_M5 y BD_MT_S2_M5	-0.008
30	BI_MT_S2_M6 y BD_MT_S2_M6	-0.101
31	BI_MT_S2_M6 y BD_MT_S2_M6	-0.059
32	BI_MT_S2_M8 y BD_MT_S2_M8	-0.002
33	BI_MBI_S2_M1 y BD_MBI_S2_M1	0.041
34	BI_MBI_S2_M2 y BD_MBI_S2_M2	0.313
35	BI_MBI_S2_M3 y BD_MBI_S2_M3	0.093
36	BI_MBI_S2_M4 y BD_MBI_S2_M4	0.036
37	BI_MBI_S2_M5 y BD_MBI_S2_M5	-0.261
38	BI_MBI_S2_M6 y BD_MBI_S2_M6	0.309
39	BI_MBI_S2_M7 y BD_MBI_S2_M7	-0.096
40	BI_MBI_S2_M8 y BD_MBI_S2_M8	-0.089
41	BI_MBS_S2_M1 y BD_MBS_S2_M1	0.118

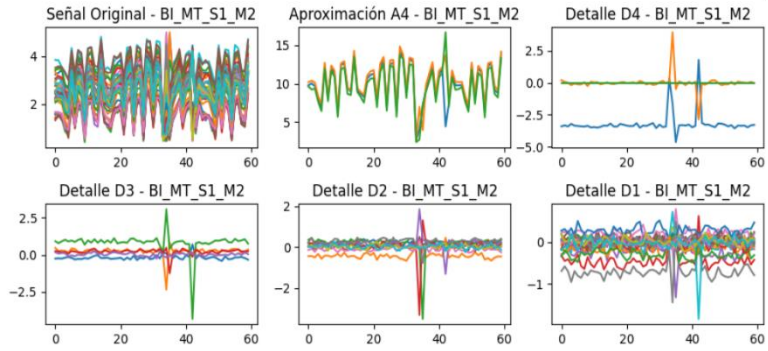
42	BI_MBS_S2_M2 y BD_MBS_S2_M2	0.063
43	BI_MBS_S2_M3 y BD_MBS_S2_M3	0.121
44	BI_MBS_S2_M4 y BD_MBS_S2_M4	0.198
45	BI_MBS_S2_M5 y BD_MBS_S2_M5	0.199
46	BI_MBS_S2_M6 y BD_MBS_S2_M6	0.285
47	BI_MBS_S2_M7 y BD_MBS_S2_M7	-0.154
48	BI_MBS_S2_M8 y BD_MBS_S2_M8	-0.140
49	BI_MT_S3_M1 y BD_MT_S3_M1	-0.148
50	BI_MT_S3_M2 y BD_MT_S3_M2	-0.181
51	BI_MT_S3_M3 y BD_MT_S3_M3	0.019
52	BI_MT_S3_M4 y BD_MT_S3_M4	-0.147
53	BI_MT_S3_M5 y BD_MT_S3_M5	-0.028
54	BI_MT_S3_M6 y BD_MT_S3_M6	-0.042
55	BI_MT_S3_M7 y BD_MT_S3_M7	-0.042
56	BI_MT_S3_M8 y BD_MT_S3_M8	-0.067
57	BI_MBI_S3_M1 y BD_MBI_S3_M1	0.198
58	BI_MBI_S3_M2 y BD_MBI_S3_M2	-0.145
59	BI_MBI_S3_M3 y BD_MBI_S3_M3	-0.077
60	BI_MBI_S3_M4 y BD_MBI_S3_M4	0.128
61	BI_MBI_S3_M5 y BD_MBI_S3_M5	-0.150
62	BI_MBI_S3_M6 y BD_MBI_S3_M6	0.112
63	BI_MBI_S3_M7 y BD_MBI_S3_M7	0.057
64	BI_MBI_S3_M8 y BD_MBI_S3_M8	0.051
65	BI_MBS_S3_M1 y BD_MBS_S3_M1	0.195
66	BI_MBS_S3_M2 y BD_MBS_S3_M2	0.022
67	BI_MBS_S3_M3 y BD_MBS_S3_M3	-0.187
68	BI_MBS_S3_M3 y BD_MBS_S3_M3	0.087
69	BI_MBS_S3_M5 y BD_MBS_S3_M5	-0.107
70	BI_MBS_S3_M6 y BD_MBS_S3_M6	-0.074
71	BI_MBS_S3_M7 y BD_MBS_S3_M7	-0.188
72	BI_MBS_S3_M8 y BD_MBS_S3_M8	-0.015

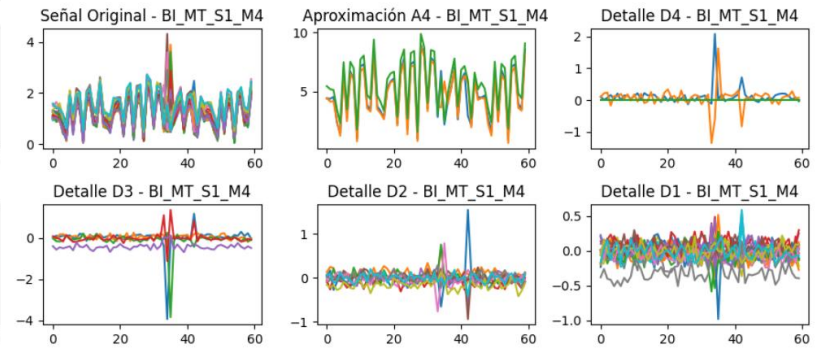
Realizado por: Vaca-Paguay, 2024.

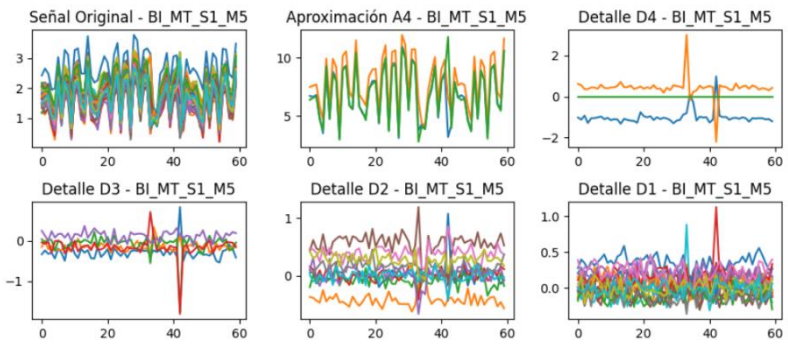
ANEXO I: ANÁLISIS DE ONDÍCULAS O WAVELET

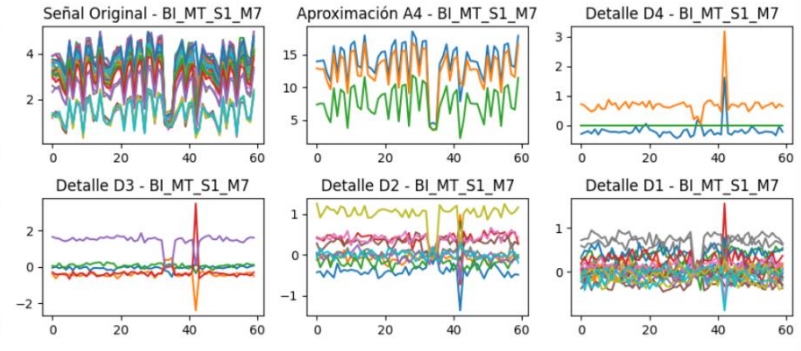


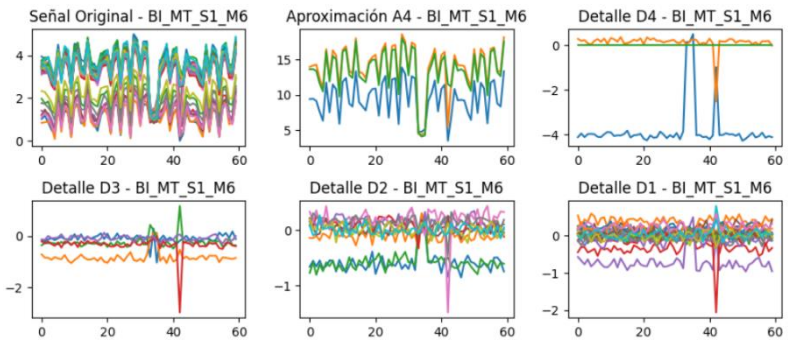


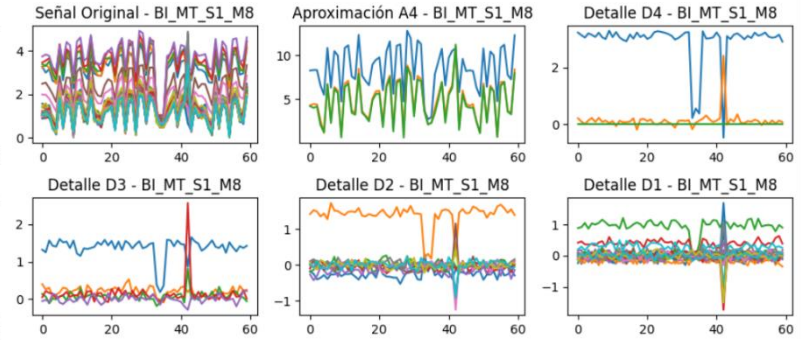


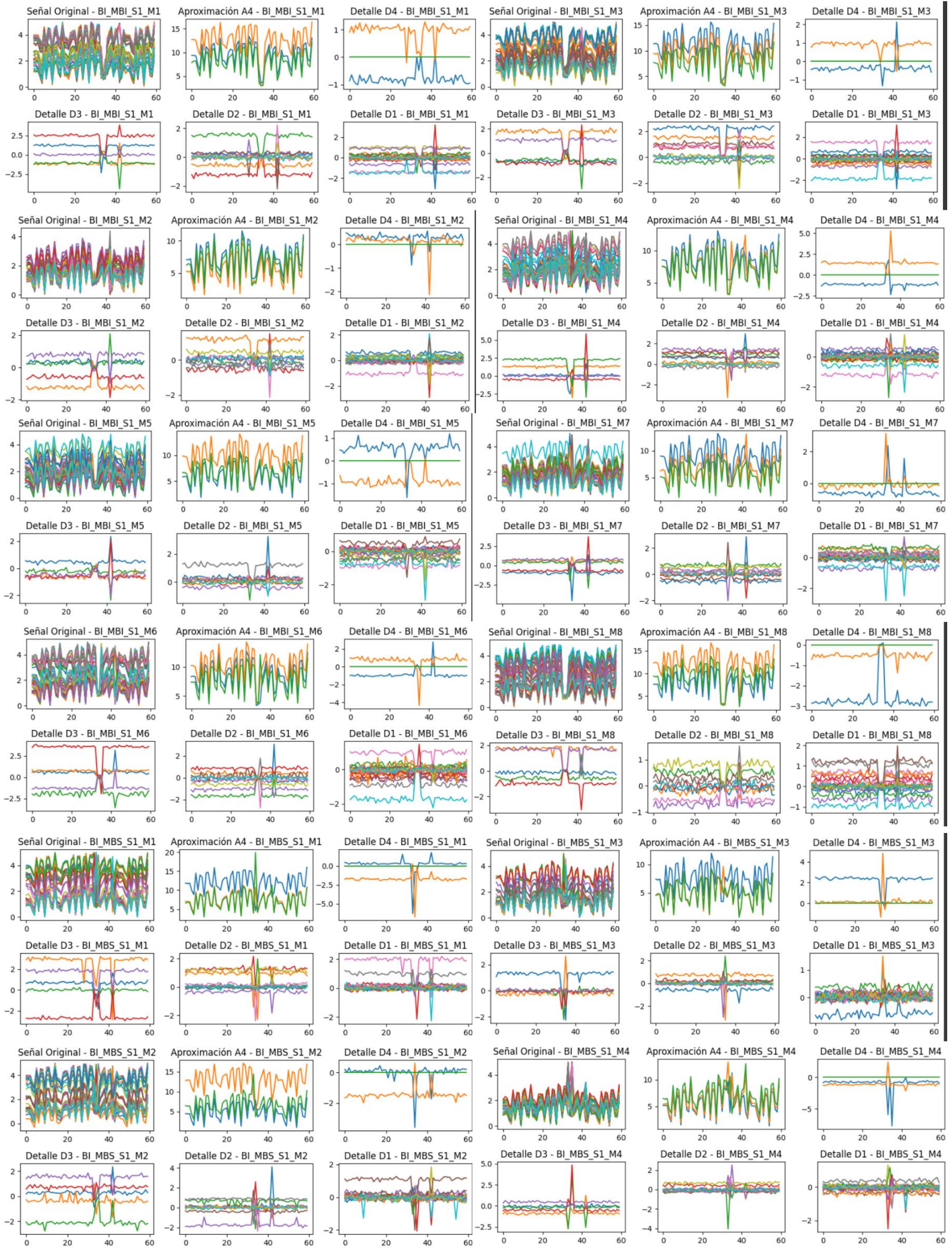


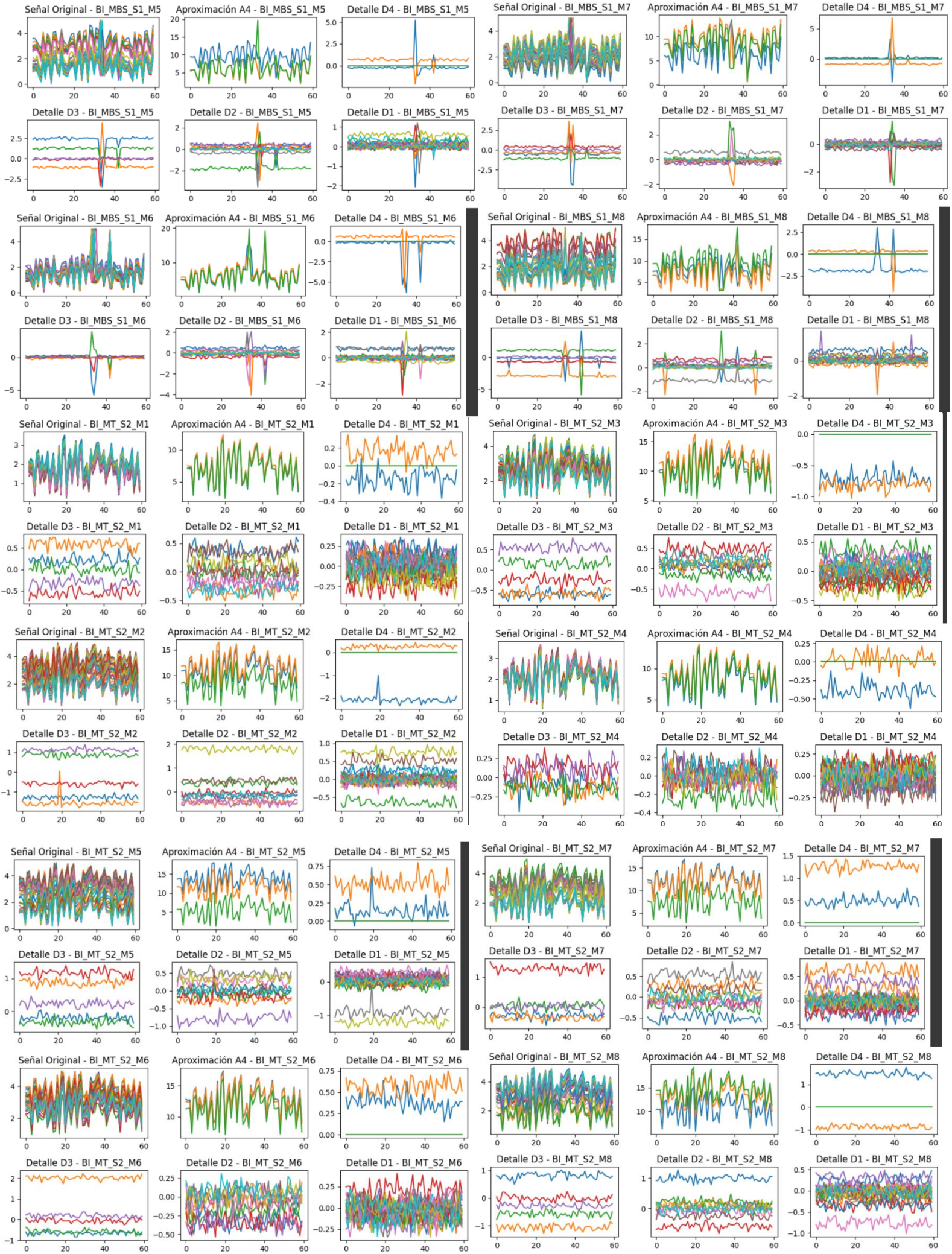









Señal Original - BI_MBI_S1_M1
Aproximación A4 - BI_MBI_S1_M1
Detalle D4 - BI_MBI_S1_M1
Señal Original - BI_MBI_S1_M3
Aproximación A4 - BI_MBI_S1_M3
Detalle D4 - BI_MBI_S1_M3
Detalle D3 - BI_MBI_S1_M1
Detalle D2 - BI_MBI_S1_M1
Detalle D1 - BI_MBI_S1_M1
Detalle D3 - BI_MBI_S1_M3
Detalle D2 - BI_MBI_S1_M3
Detalle D1 - BI_MBI_S1_M3
Señal Original - BI_MBI_S1_M2
Aproximación A4 - BI_MBI_S1_M2
Detalle D4 - BI_MBI_S1_M2
Señal Original - BI_MBI_S1_M4
Aproximación A4 - BI_MBI_S1_M4
Detalle D4 - BI_MBI_S1_M4
Detalle D3 - BI_MBI_S1_M2
Detalle D2 - BI_MBI_S1_M2
Detalle D1 - BI_MBI_S1_M2
Detalle D3 - BI_MBI_S1_M4
Detalle D2 - BI_MBI_S1_M4
Detalle D1 - BI_MBI_S1_M4
Señal Original - BI_MBI_S1_M5
Aproximación A4 - BI_MBI_S1_M5
Detalle D4 - BI_MBI_S1_M5
Señal Original - BI_MBI_S1_M7
Aproximación A4 - BI_MBI_S1_M7
Detalle D4 - BI_MBI_S1_M7
Detalle D3 - BI_MBI_S1_M5
Detalle D2 - BI_MBI_S1_M5
Detalle D1 - BI_MBI_S1_M5
Detalle D3 - BI_MBI_S1_M7
Detalle D2 - BI_MBI_S1_M7
Detalle D1 - BI_MBI_S1_M7
Señal Original - BI_MBI_S1_M6
Aproximación A4 - BI_MBI_S1_M6
Detalle D4 - BI_MBI_S1_M6
Señal Original - BI_MBI_S1_M8
Aproximación A4 - BI_MBI_S1_M8
Detalle D4 - BI_MBI_S1_M8
Detalle D3 - BI_MBI_S1_M6
Detalle D2 - BI_MBI_S1_M6
Detalle D1 - BI_MBI_S1_M6
Detalle D3 - BI_MBI_S1_M8
Detalle D2 - BI_MBI_S1_M8
Detalle D1 - BI_MBI_S1_M8
Señal Original - BI_MBS_S1_M1
Aproximación A4 - BI_MBS_S1_M1
Detalle D4 - BI_MBS_S1_M1
Señal Original - BI_MBS_S1_M3
Aproximación A4 - BI_MBS_S1_M3
Detalle D4 - BI_MBS_S1_M3
Detalle D3 - BI_MBS_S1_M1
Detalle D2 - BI_MBS_S1_M1
Detalle D1 - BI_MBS_S1_M1
Detalle D3 - BI_MBS_S1_M3
Detalle D2 - BI_MBS_S1_M3
Detalle D1 - BI_MBS_S1_M3
Señal Original - BI_MBS_S1_M2
Aproximación A4 - BI_MBS_S1_M2
Detalle D4 - BI_MBS_S1_M2
Señal Original - BI_MBS_S1_M4
Aproximación A4 - BI_MBS_S1_M4
Detalle D4 - BI_MBS_S1_M4
Detalle D3 - BI_MBS_S1_M2
Detalle D2 - BI_MBS_S1_M2
Detalle D1 - BI_MBS_S1_M2
Detalle D3 - BI_MBS_S1_M4
Detalle D2 - BI_MBS_S1_M4
Detalle D1 - BI_MBS_S1_M4


Señal Original - BI_MBS_S1_M5
Aproximación A4 - BI_MBS_S1_M5
Detalle D4 - BI_MBS_S1_M5
Señal Original - BI_MBS_S1_M7
Aproximación A4 - BI_MBS_S1_M7
Detalle D4 - BI_MBS_S1_M7
Detalle D3 - BI_MBS_S1_M5
Detalle D2 - BI_MBS_S1_M5
Detalle D1 - BI_MBS_S1_M5
Detalle D3 - BI_MBS_S1_M7
Detalle D2 - BI_MBS_S1_M7
Detalle D1 - BI_MBS_S1_M7
Señal Original - BI_MBS_S1_M6
Aproximación A4 - BI_MBS_S1_M6
Detalle D4 - BI_MBS_S1_M6
Señal Original - BI_MBS_S1_M8
Aproximación A4 - BI_MBS_S1_M8
Detalle D4 - BI_MBS_S1_M8
Detalle D3 - BI_MBS_S1_M6
Detalle D2 - BI_MBS_S1_M6
Detalle D1 - BI_MBS_S1_M6
Detalle D3 - BI_MBS_S1_M8
Detalle D2 - BI_MBS_S1_M8
Detalle D1 - BI_MBS_S1_M8
Señal Original - BI_MT_S2_M1
Aproximación A4 - BI_MT_S2_M1
Detalle D4 - BI_MT_S2_M1
Señal Original - BI_MT_S2_M3
Aproximación A4 - BI_MT_S2_M3
Detalle D4 - BI_MT_S2_M3
Detalle D3 - BI_MT_S2_M1
Detalle D2 - BI_MT_S2_M1
Detalle D1 - BI_MT_S2_M1
Detalle D3 - BI_MT_S2_M3
Detalle D2 - BI_MT_S2_M3
Detalle D1 - BI_MT_S2_M3
Señal Original - BI_MT_S2_M2
Aproximación A4 - BI_MT_S2_M2
Detalle D4 - BI_MT_S2_M2
Señal Original - BI_MT_S2_M4
Aproximación A4 - BI_MT_S2_M4
Detalle D4 - BI_MT_S2_M4
Detalle D3 - BI_MT_S2_M2
Detalle D2 - BI_MT_S2_M2
Detalle D1 - BI_MT_S2_M2
Detalle D3 - BI_MT_S2_M4
Detalle D2 - BI_MT_S2_M4
Detalle D1 - BI_MT_S2_M4
Señal Original - BI_MT_S2_M5
Aproximación A4 - BI_MT_S2_M5
Detalle D4 - BI_MT_S2_M5
Señal Original - BI_MT_S2_M7
Aproximación A4 - BI_MT_S2_M7
Detalle D4 - BI_MT_S2_M7
Detalle D3 - BI_MT_S2_M5
Detalle D2 - BI_MT_S2_M5
Detalle D1 - BI_MT_S2_M5
Detalle D3 - BI_MT_S2_M7
Detalle D2 - BI_MT_S2_M7
Detalle D1 - BI_MT_S2_M7
Señal Original - BI_MT_S2_M6
Aproximación A4 - BI_MT_S2_M6
Detalle D4 - BI_MT_S2_M6
Señal Original - BI_MT_S2_M8
Aproximación A4 - BI_MT_S2_M8
Detalle D4 - BI_MT_S2_M8
Detalle D3 - BI_MT_S2_M6
Detalle D2 - BI_MT_S2_M6
Detalle D1 - BI_MT_S2_M6
Detalle D3 - BI_MT_S2_M8
Detalle D2 - BI_MT_S2_M8
Detalle D1 - BI_MT_S2_M8

ANEXO J: CORRELACIONES ENTRE SEÑALES ORIGINALES Y PREDICCIONES DEL MODELO

Data Frame BI	Correlación	Data Frame BD	Correlación
BI_MT_S1_M1	0.983	BD_MT_S1_M1	0.949
BI_MT_S1_M2	0.982	BD_MT_S1_M2	0.98
BI_MT_S1_M3	0.988	BD_MT_S1_M3	0.967
BI_MT_S1_M4	0.973	BD_MT_S1_M4	0.974
BI_MT_S1_M5	0.977	BD_MT_S1_M5	0.979
BI_MT_S1_M6	0.982	BD_MT_S1_M6	0.974
BI_MT_S1_M7	0.975	BD_MT_S1_M7	0.97
BI_MT_S1_M8	0.975	BD_MT_S1_M8	0.981
BI_MBI_S1_M1	0.982	BD_MBI_S1_M1	0.961
BI_MBI_S1_M2	0.973	BD_MBI_S1_M2	0.957
BI_MBI_S1_M3	0.96	BD_MBI_S1_M3	0.966
BI_MBI_S1_M4	0.944	BD_MBI_S1_M4	0.972
BI_MBI_S1_M5	0.972	BD_MBI_S1_M5	0.981
BI_MBI_S1_M6	0.985	BD_MBI_S1_M6	0.982
BI_MBI_S1_M7	0.976	BD_MBI_S1_M7	0.978
BI_MBI_S1_M8	0.981	BD_MBI_S1_M8	0.99
BI_MBS_S1_M1	0.989	BD_MBS_S1_M1	0.96
BI_MBS_S1_M2	0.975	BD_MBS_S1_M2	0.972
BI_MBS_S1_M3	0.982	BD_MBS_S1_M3	0.984
BI_MBS_S1_M4	0.957	BD_MBS_S1_M4	0.973
BI_MBS_S1_M5	0.986	BD_MBS_S1_M5	0.972
BI_MBS_S1_M6	0.977	BD_MBS_S1_M6	0.973
BI_MBS_S1_M7	0.988	BD_MBS_S1_M7	0.96
BI_MBS_S1_M8	0.984	BD_MBS_S1_M8	0.966
BI_MT_S2_M1	0.985	BD_MT_S2_M1	0.963
BI_MT_S2_M2	0.981	BD_MT_S2_M2	0.974
BI_MT_S2_M3	0.982	BD_MT_S2_M3	0.955
BI_MT_S2_M4	0.986	BD_MT_S2_M4	0.972
BI_MT_S2_M5	0.975	BD_MT_S2_M5	0.954
BI_MT_S2_M6	0.985	BD_MT_S2_M6	0.981
BI_MT_S2_M7	0.979	BD_MT_S2_M7	0.973
BI_MT_S2_M8	0.982	BD_MT_S2_M8	0.966
BI_MBI_S2_M1	0.986	BD_MBI_S2_M1	0.98
BI_MBI_S2_M2	0.986	BD_MBI_S2_M2	0.979
BI_MBI_S2_M3	0.979	BD_MBI_S2_M3	0.984
BI_MBI_S2_M4	0.984	BD_MBI_S2_M4	0.979
BI_MBI_S2_M5	0.987	BD_MBI_S2_M5	0.981
BI_MBI_S2_M6	0.984	BD_MBI_S2_M6	0.983
BI_MBI_S2_M7	0.982	BD_MBI_S2_M7	0.968
BI_MBI_S2_M8	0.982	BD_MBI_S2_M8	0.974

BI_MBS_S2_M1	0.986	BD_MBS_S2_M1	0.978
BI_MBS_S2_M2	0.986	BD_MBS_S2_M2	0.975
BI_MBS_S2_M3	0.98	BD_MBS_S2_M3	0.971
BI_MBS_S2_M4	0.985	BD_MBS_S2_M4	0.978
BI_MBS_S2_M5	0.987	BD_MBS_S2_M5	0.951
BI_MBS_S2_M6	0.983	BD_MBS_S2_M6	0.952
BI_MBS_S2_M7	0.982	BD_MBS_S2_M7	0.978
BI_MBS_S2_M8	0.982	BD_MBS_S2_M8	0.976
BI_MT_S3_M1	0.98	BD_MT_S3_M1	0.972
BI_MT_S3_M2	0.978	BD_MT_S3_M2	0.981
BI_MT_S3_M3	0.977	BD_MT_S3_M3	0.985
BI_MT_S3_M4	0.989	BD_MT_S3_M4	0.97
BI_MT_S3_M5	0.987	BD_MT_S3_M5	0.985
BI_MT_S3_M6	0.991	BD_MT_S3_M6	0.988
BI_MT_S3_M7	0.989	BD_MT_S3_M7	0.988
BI_MT_S3_M8	0.992	BD_MT_S3_M8	0.982
BI_MBI_S3_M1	0.937	BD_MBI_S3_M1	0.978
BI_MBI_S3_M2	0.893	BD_MBI_S3_M2	0.984
BI_MBI_S3_M3	0.962	BD_MBI_S3_M3	0.963
BI_MBI_S3_M4	0.983	BD_MBI_S3_M4	0.976
BI_MBI_S3_M5	0.975	BD_MBI_S3_M5	0.984
BI_MBI_S3_M6	0.963	BD_MBI_S3_M6	0.98
BI_MBI_S3_M7	0.982	BD_MBI_S3_M7	0.985
BI_MBI_S3_M8	0.988	BD_MBI_S3_M8	0.985
BI_MBS_S3_M1	0.986	BD_MBS_S3_M1	0.982
BI_MBS_S3_M2	0.981	BD_MBS_S3_M2	0.991
BI_MBS_S3_M3	0.975	BD_MBS_S3_M3	0.982
BI_MBS_S3_M4	0.95	BD_MBS_S3_M4	0.986
BI_MBS_S3_M5	0.974	BD_MBS_S3_M5	0.987
BI_MBS_S3_M6	0.939	BD_MBS_S3_M6	0.986
BI_MBS_S3_M7	0.98	BD_MBS_S3_M7	0.988
BI_MBS_S3_M8	0.967	BD_MBS_S3_M8	0.985

Realizado por: Vaca-Paguay, 2024