UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA MECÁNICA

TESIS

Diseño de un sistema de control mioeléctrico con clasificación en tiempo real de movimientos manuales basado en Deep Learning para mejorar la destreza de las prótesis de mano de agarre simple

PARA OBTENER EL TÍTULO PROFESIONAL DE:

INGENIERO MECÁTRONICO

ELABORADO POR

EDGARD JESÚS BARAZORDA RODRÍGUEZ

0000-0002-7698-0469

ASESOR

Msc. Ing. DANIEL LEONARDO BARRERA ESPARTA

0000-0003-3465-788X

LIMA – PERÚ

2023

Dedicatoria

Dedico esta tesis a mis padres por ser mis ejemplos a seguir y mis guías para no rendirme. Asimismo, dedico este trabajo a mi abuela y hermanas por sus constantes consideraciones y apoyos.

Agradecimiento

Agradezco a la Universidad Nacional de Ingeniería, profesores y compañeros de estudios, por sus contribuciones en mi formación profesional. Especialmente, al Msc. Ing. Daniel Barrera, quien me brindó su apoyo y conocimiento en cada etapa de la presente investigación.

RESUMEN

En el Perú, las prótesis de mano con alta destreza son adquiridas casi exclusivamente mediante la importación, lo que conlleva altos costos y tiempos de espera. Asimismo, los desarrollos nacionales en el tema han tenido un enfoque de control de agarre simple, limitando el potencial que puede ofrecer el diseño mecánico. Con el fin de aumentar la destreza de este tipo de prótesis, el trabajo de investigación pretende diseñar un sistema de control mioeléctrico con clasificación en tiempo real de movimientos manuales basado en Deep Learning. El sistema de control propuesto consta de 3 subsistemas: adquisición, para capturar señales electromiografías (EMG) de 5 movimientos manuales relacionadas a las actividades de la vida diaria (AVD); procesamiento, para estandarizar, filtrar y convertir las señales en imágenes (escalogramas) utilizando la Transformada Wavelet Compleja (CWT); y clasificación, para entrenar un modelo CNN *MobileNet* V2 tomando los escalogramas como entrada. Con el fin de mejorar la interacción con el sistema de control, se programó una GUI que adquiera, procese y clasifique las señales en tiempo real; asimismo, para generar una retroalimentación visual en los usuarios se simuló una prótesis de código abierto. Mediante las pruebas experimentales realizadas con 5 voluntarios se obtuvo una precisión media de clasificación en tiempo real de 87.3% y una latencia de control media de 276.97 ms. Estos resultados respaldan una mejora del 30% en la destreza en comparación con los sistemas convencionales de control de agarre simple.

Palabras claves: EMG, Deep Learning, Control en tiempo real, Prótesis, GUI

ABSTRACT

In Peru, highly dexterous hand prostheses are primarily acquired through imports, incurring significant costs and lengthy waiting times. Additionally, domestic developments in this field have predominantly focused on simple grip control, thereby limiting the potential of mechanical design. To enhance the dexterity of such prostheses, this research endeavors to design a myoelectric control system with real-time classification of manual movements based on Deep Learning. The proposed control system comprises three subsystems: acquisition, to capture electromyographic (EMG) signals related to five manual movements associated with activities of daily living (ADL); processing, for standardizing, filtering, and converting these signals into images (spectrograms) using Complex Wavelet Transform (CWT); and classification, to train a CNN MobileNet V2 model utilizing spectrograms as input. To enhance user interaction, a Graphical User Interface (GUI) was programmed to acquire, process, and classify signals in real-time. Furthermore, an open-source prosthesis was simulated to provide visual feedback to users. Experimental tests conducted with five volunteers yielded an average real-time classification accuracy of 87.3% and an average control latency of 276.97 ms. These results substantiate a 30% enhancement in dexterity compared to conventional simple grip control systems.

Keywords: EMG, Deep Learning, Real-time control, Prosthesis, GUI

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

PRÓLOGO

Las personas con algún tipo de discapacidad motora manual usualmente están limitadas para realizar sus actividades de la vida diaria con normalidad, por ejemplo, cocinar, limpiar, trabajar, entre otras. En este contexto, las prótesis de mano de alta destreza surgen como una solución moderna para restaurar parcial o totalmente la función de sujetar y manipular objetos con agilidad. Sin embargo, en el Perú todavía la industria de este tipo de prótesis no ha sido completamente desarrollada, siendo la importación la alternativa más viable para su adquisición a pesar de los altos costos y tiempos de espera. Es por ello que el presente trabajo busca aportar un diseño de un sistema de control mioeléctrico con clasificación en tiempo real de movimientos manuales basado en Deep Learning para mejorar la destreza de las prótesis de mano de agarre simple.

En el capítulo I se describen las generalidades y el problema que aborda la investigación; asimismo, se definen los objetivos e hipótesis del estudio. Y para una mejor contextualización en la clasificación de señales musculares, se mencionan antecedentes investigativos a nival internacional, nacional y local.

En el capítulo II se desarrolla el marco teórico y conceptual para un mayor entendimiento del trabajo de investigación, enfatizando la teoría relacionada con la electromiografía; procesamiento y clasificación de señales musculares; Interfaz Gráfica de Usuario; prótesis de mano y destreza.

En el capítulo III se detalla el desarrollo de la tesis, dividido en los diseños de los subsistemas de adquisición, procesamiento y clasificación de señales EMG; y de igual modo, la programación de una GUI y la simulación de una prótesis de mano de código abierto. El objetivo de este capítulo es diseñar un sistema de control que adquiera, procese y clasifique en tiempo real las señales de un usuario logrando una retroalimentación visual al reflejar su intención de movimiento en la simulación.

En el capítulo IV se presentan los resultados de las pruebas de destreza realizadas a 5 voluntarios para medir la precisión y latencia de control promedio de la clasificación en tiempo real de los movimientos manuales, los cuales están directamente relacionados con la destreza de una prótesis; asimismo, la comparación entre el sistema de control propuesto y los sistemas de control de agarre simple convencionales.

Finalmente, se enuncian las conclusiones sustentadas en las pruebas experimentales y las recomendaciones para favorecer la mejora del diseño propuesto.

CAPÍTULO I INTRODUCCIÓN

1.1. Generalidades

Según el informe del Banco Mundial titulado *"World Report on Disability"*, el número de personas que viven con algún tipo de discapacidad es aproximadamente mil millones. Estas personas a menudo enfrentan barreras significativas en el acceso a servicios esenciales como la sanidad, la educación y el trabajo, lo que limita su participación activa en la sociedad. Además, el informe destaca la relación directa entre la discapacidad y las altas tasas de pobreza con una baja esperanza de vida (Organización Mundial de la Salud, 2011).

En el Perú el número de personas que experimentan dificultades o limitaciones para llevar a cabo sus actividades diarias ha superado los tres millones; siendo las dificultades motoras una de las discapacidades más comunes, afectando a 485 mil 211 personas. Una quinta parte de esta población sufre limitaciones en su vida cotidiana, como por ejemplo no poder utilizar cubiertos para alimentarse o agarrar objetos, debido a una amputación o malformación (INEI, 2017).

El uso de prótesis es crucial para permitir que las personas con deficiencias físicas o limitaciones funcionales vivan de manera independiente y participen plenamente en la sociedad. Estas ayudas pueden reducir la necesidad de apoyos formales, tratamientos y cuidadores, evitando así la exclusión y el aislamiento social, y disminuyendo la morbilidad y la discapacidad (González, 2020).

Sin embargo, en el Perú la fabricación y la comercialización de prótesis están enfocadas en las prótesis estéticas o de agarre simple las cuales tienen una baja destreza en el desarrollo de las actividades de la vida diaria (AVD). Las prótesis de alta destreza que brindan un mayor control y precisión a los usuarios, tienen un alto costo y son mayormente importadas (Soto Bustamante, 2016).

El trabajo de investigación presente pretende plantear una solución para mejorar la destreza de las prótesis de mano de agarre simple en el contexto peruano mediante el diseño de un sistema de control mioléctrico, el cual permita a un usuario controlar en tiempo de real una prótesis al clasificar 5 movimientos manuales relacionadas a las AVD, utilizando algoritmos de Deep Learning. Al diseñar el sistema propuesto se espera contribuir en los avances de la destreza de las prótesis, empoderando y mejorando la independencia de las personas con alguna amputación o deficiencia motriz en la mano.

1.2. Descripción del Problema de Investigación

El estudio Carga Mundial de Morbilidad, estima que el 19.4% de la población mayor a 15 años tiene algún tipo de discapacidad. En muchos casos, la discapacidad no permite que las personas gocen de un buen nivel de salud física y que tengan un mayor riesgo que las personas sin discapacidad de desarrollar depresión o condiciones de salud crónicas como diabetes, estenosis arterial o enfermedades cardiovasculares (Organización Mundial de la Salud, 2011).

Según la Organización para la Cooperación y Desarrollo Económicos (OCDE) la tasa de discapacidad en personas con menos educación es en promedio 19%, en comparación del 11% de las personas que tienen más educación. Por otra parte, en el ámbito laboral, la Encuesta Mundial de Salud realizada a 51 países estima que los porcentajes de ocupación de varones y mujeres con discapacidad son del 52.8% y 19.6% respectivamente, en comparación del 64.9% y 29.9% de las varones y mujeres sin discapacidad. Esta situación es debido en parte a la discriminación laboral generalizada que padecen las personas con discapacidad; ello se refleja en los bajos ingresos y en la falta de empleo. (Organización Mundial de la Salud, 2011).

En América Latina, las crisis económicas y el impacto de la pandemia del COVID-19 han acelerado el aumento de la vulnerabilidad de las personas discapacitadas. En la actualidad, 1 de cada 5 hogares en pobreza extrema viven con una persona con discapacidad; asimismo, 7 de cada 10 hogares con al menos una persona

discapacitada son altamente probables a caer en la pobreza. El escenario en esta región solo pronostica un imparable incremento en la población con discapacidad (Garcia Mora et al., 2021).

El censo de población y vivienda realizado el 2017 por el INEI en el Perú detalla que solamente el 18.4% de niños con discapacidad asisten a algún centro de enseñanza, en comparación del 35.4% de niños que no tienen discapacidad. Asimismo, solo el 39.6% de los peruanos con discapacidad integra la Población Económicamente Activa (PEA), lo que significa 22.3 puntos porcentuales menos que la población sin discapacidad (INEI, 2017).

El Instituto Nacional de Estadística e Informática (INEI, 2017) mostró que más de 3 millones de peruanos sufren por lo menos alguna limitación que no les permite desarrollar sus actividades diarias con normalidad, por ejemplo, estudiar o trabajar. Dentro de las discapacidades más frecuentes en el Perú, en según lugar se ubican las dificultades motoras que abarcan la capacidad de caminar, usar los brazos, usar las piernas, que afecta a 485 mil 211 personas.

Según la primera edición de la Encuesta Nacional Especializada sobre Discapacidad (ENEDIS), el 21.5% de las personas con discapacidad de locomoción y/o destreza presenta dificultades para manipular herramientas manuales; por ejemplo, los utensilios de cocina o las tijeras. Mientras tanto, el 19.1% presenta dificultades para agarrar objetos pequeños cotidianos como las monedas (INEI, 2012).

En el estudio (Camacho Conchucos, 2010) del Instituto Nacional de Rehabilitación (INR) revisó 1290 historias clínicas de pacientes amputados donde el 8.4% (108) fueron causados por accidente de trabajo, lo que significó una pérdida de 14.5 años de productividad en media, o 1569 años en total. El estudio concluye que, a pesar de ser una población estudiada, aproximadamente la mitad no podrá desempeñar su ocupación, generando una carga económica familiar.

Si bien las prótesis son de ayuda para que una persona con un grado amputación pueda recuperar su autonomía para poder realizar sus actividades, aún existe una escasez en la producción de prótesis de mano funcionales en el Perú. Una prótesis estética en un material de PVC tiene un costo de 1000 soles, mientras que una prótesis en el material de silicona puede llegar a costar 1200 dólares (Bustamante Carvallo, 2018). Por otra parte, el costo de una prótesis mecánica de agarre simple puede llegar a 4000 soles; y una prótesis mioeléctrica comercial con una mayor destreza, a 25000 soles (PERU21, 2021).

La producción de equipamientos médicos en el Perú está en un segundo plano (incluyendo las prótesis), por lo que prima las importaciones desde países como Estados Unidos y Suiza con un alto precio y gran tiempo de espera (Soto Bustamante, 2016).

En este contexto, es notorio la necesidad de mejorar la destreza de las prótesis de mano en el Perú, especialmente las de agarre simple, mediante un sistema de control basado en tecnologías potenciales como la Inteligencia Artificial.

1.3. Objetivos del Estudio

1.3.1. Objetivo General

- Diseñar un sistema de control mioeléctrico con clasificación en tiempo real de movimientos manuales basado en Deep Learning para mejorar la destreza de las prótesis de mano de agarre simple.

1.3.2. Objetivos Específicos

- Seleccionar técnicas de adquisición y procesamiento de señales EMG para obtener escalogramas favorables en el entrenamiento de modelos de clasificación de movimientos manuales.
- Comparar clasificadores a partir de escalogramas para estimar en tiempo real la intención de movimiento manual.
- Validar el sistema de control mediante pruebas de destreza con la simulación de una prótesis conectada a una GUI de señales EMG.

1.4. Hipótesis

1.4.1. Hipótesis General

- El diseño de un sistema de control mioeléctrico con clasificación en tiempo real de movimientos manuales basado en Deep Learning mejorará la destreza de las prótesis de agarre simple.

1.4.2. Hipótesis Específicas

- La selección de técnicas de adquisición y procesamiento de señales ayudará a obtener escalogramas favorables en el entrenamiento de modelos de clasificación de movimientos manuales.
- La comparación de clasificadores a partir de escalogramas estimará en tiempo real la intención de movimiento manual.

- La validación del sistema de control será posible a través de pruebas de destreza con la simulación de una prótesis de mano conectado a una GUI de señales EMG.

1.5. Antecedentes Investigativos

A continuación, se describen los antecedentes investigativos en relación a la tesis en el ámbito internacional, nacional y local.

1.5.1. Ámbito Internacional

En el trabajo de (Fazeli et al., 2020) se fabricó un electrodo seco y flexible para demostrar su calidad de adquisición de señales EMG. Durante su investigación se probó su funcionamiento con 2 modelos de máquina de vectores de soporte (SVM) gaussiano con el objetivo de clasificar 5 movimientos manuales entrenados mediante bases de datos de electrodos EMG secos y húmedos comerciales. Los resultados mencionan que las precisiones utilizando el electrodo fabricado para 3 sujetos con el modelo del electrodo seco comercial fueron de 96.97%, 99%, 97.34%, y con el modelo del electrodo húmedo comercial fueron de 98.3%, 98.67% y 97.67%.

En el trabajo de (Lu et al., 2020) se propuso un innovativo método para la identificación personal de 21 sujetos basado en la clasificación de señales electromiográficas mediante el brazalete *Myo* en la articulación radiohumeral cuando se realiza el gesto de mano abierta. Para el procesamiento y clasificación de las señales EMG se propuso un método utilizando la Transformada Wavelet Discreta (DWT) con ExtraTreesClassifier, y un método utilizando la Transformada

Wavelet Continua (CWT) con Redes Neuronales Convolucionales (CNN) más *Transfer Learning*, logrando el 99,206 % y el 99,203 % de precisión de identificación personal respectivamente (ver figura 1.1).

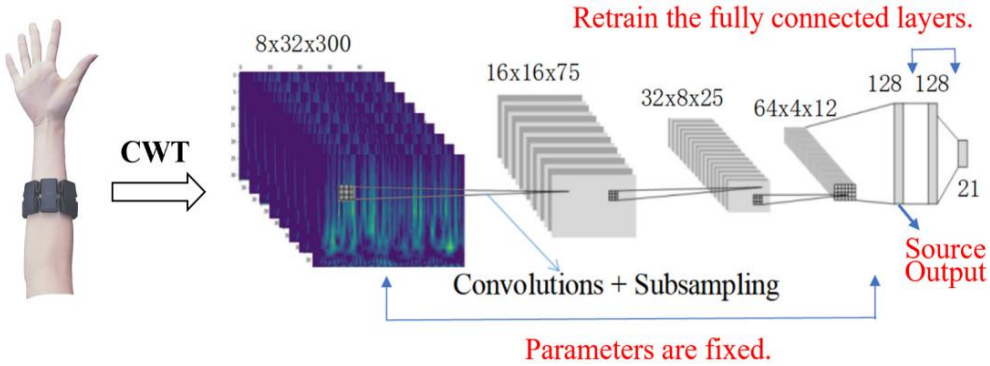


Figura 1.1. Estructura de la identificación personal.
Fuente: Lu et al. (2020) Structure of personal identification

En el trabajo de (Oh & Jo, 2019) se propuso un clasificador de gestos manuales a partir de bases de datos de tres gestos de agarre y tres gestos de lenguaje de señas adquiridas con ochos sensores EMG de superficie del brazalete comercial *Myo*. Los resultados demostraron que mediante la transformada Walevet (WT) y la transformada Walevet promedio a escala (SAWT) se pudo generar imágenes para ser entrenadas por una Red Neuronal Convolucional (CNN) como se muestra en la figura 1.2. En los resultados de la investigación se obtuvo una precisión de estimación de los seis gestos manuales de 94%.

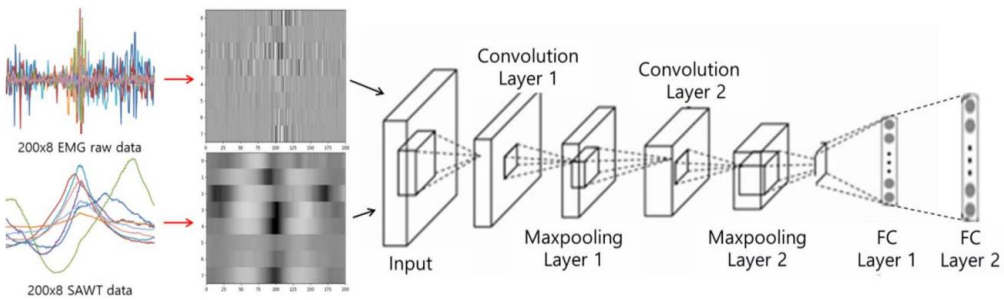


Figura 1.2. La estructura de la Red Neuronal Convolucional.
Fuente: Oh & Jo. (2019) La estructura de la Red Neuronal Convolucional

En la investigación de (Forero et al., 2022) se ha desarrollado una metodología para la simulación de prótesis transradial en un entorno virtual adquiriendo señales de electrooculagrafía (EOG) para la rehabilitación de agarre manual. En primer lugar, se diseñó el mecanismo de movimiento interfalángicos de los dedos y la prótesis utilizando el software SolidWorks®. Posteriormente, se ensambló la prótesis en un entorno de VR como Webots®; y, por otro lado, se realizó la adquisición y procesamiento de señales EOG mediante un circuito electrónico conectado al software Matlab®. Finalmente, se realizó la simulación de terapia de agarre mediante la conexión de Webots® con Matlab® para objetos cilíndricos, prismáticos y circulares.

1.5.2. Ámbito Nacional

En el trabajo de (Galarza Flores, 2018) se propuso una técnica de identificación de cuatro movimientos manuales: extensión, flexión, abrir y cerrar la mano. Se siguió el diagrama de la figura 1.3. En primer lugar, se adquirió de señales EMG del antebrazo mediante dos sensores comerciales *MyoWare* en los músculos braquiorradial y flexor cubital. Las señales adquiridas fueron procesadas para obtener vectores característicos que sirvan como entradas para su estimación utilizando Machine Learning supervisado. La identificación de movimientos de la mano obtuvo hasta una precisión de 91.00% considerando como entrada los coeficientes de aproximación de la Transformada Wavelet (TW) y características estadísticas de chi cuadrado.

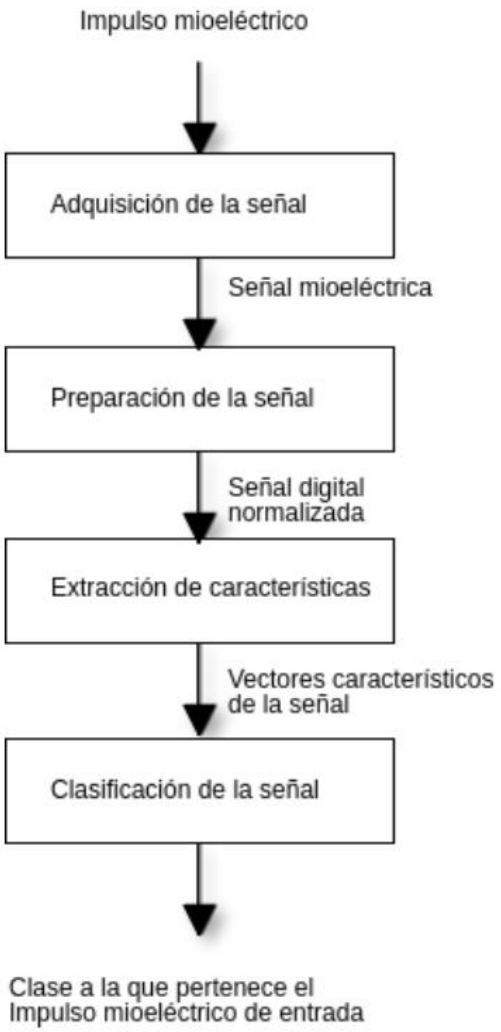


Figura 1.3. Diagrama del proceso.
Fuente: Galarza Flores. (2018) Diagrama del proceso

En el trabajo de (Briones Manrique, 2020) se presentó un sistema portátil de bajo costo para el monitoreo y adquisición de señales EMG a través de electrodos no invasivos con capacidad de conexión inalámbrica con computadoras y celulares. El sistema se logró validar adecuadamente mediante pruebas realizadas en el Hospital Nacional Arzobispo Loayza y en el instituto Nacional de Ciencia Neurológicas, al comparar las señales EMG de los voluntarios con patrones de electromiografía.

En el trabajo de (Alvarado Castillo et al., 2019) se probó dos metodologías para la adquisición de señales EMG para establecer el más efectivo en el control de una prótesis robótica de extremidad superior en siete voluntarios sanos. En la primera metodología se empleó electrodos húmedos superficiales en el músculo palma mayor, mientras que en la segunda metodología se empleó electrodos secos superficiales en el músculo flexor común superficial de los dedos. Los resultados demostraron que la disminución de impedancia de los electrodos secos no impactó en el control en virtud de la adaptabilidad de los usuarios a los nuevos niveles de señales EMG en el tiempo. En los resultados mencionan que el método con electrodos húmedos solo logró un 57.14% del control total a comparación del método con electrodos secos con un 85.71% de precisión.

En el trabajo de (Bohórquez Bendezú, 2021) se diseñó un sistema para controlar una prótesis transfemoral con base en la estimación de la intención de tres movimientos asociados a la rodilla: marcha, proceso de bipedestación y proceso de sedestación. Se empleó una base de datos a partir de la adquisición de cuatros sensores EMG ubicados en músculos de las extremidades inferiores en sujetos hombres, donde se obtuvieron 44 características en el dominio del tiempo (11 por cada señal EMG) para entrenar un modelo de Red Neuronal como se visualiza en la Figura 1.4. El sistema de reconocimiento logró un rendimiento general aceptable de 85.94% permitiendo validar el sistema de control de manera virtual.

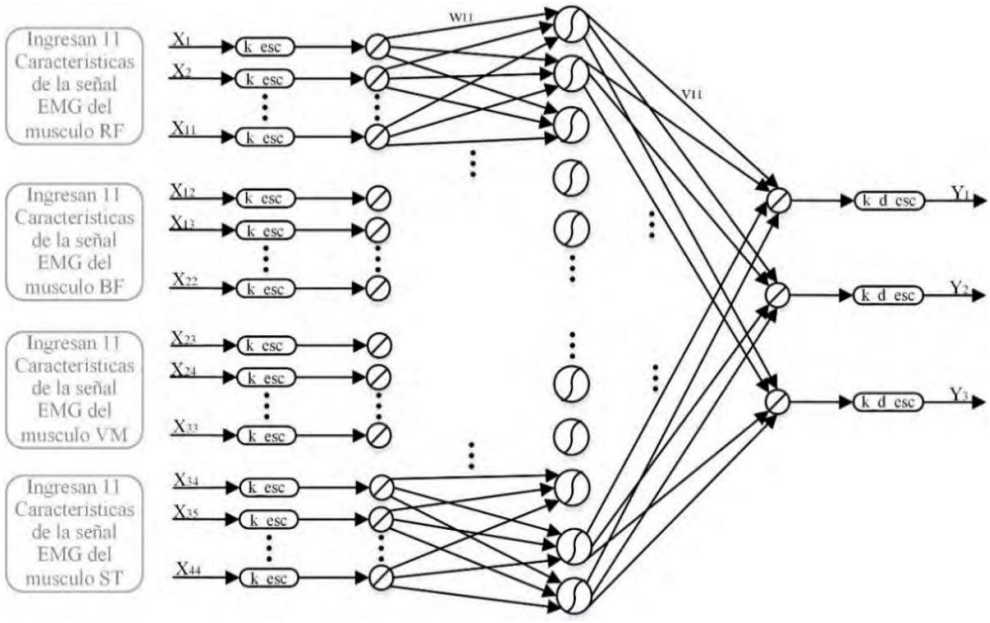


Figura 1.4. Estructura de la Red Neuronal con escalamiento.
Fuente: Bohórquez Bendezú. (2021) Estructura de la Red Neuronal con escalamiento

1.5.3. Ámbito Local

En el trabajo de (Lucas Vargas, 2021) se implementó un sistema destinado a aumentar la movilidad en entornos exteriores para las personas con discapacidad visual. Para ello se seleccionó la Red Neuronal Convolucional *MobileNet* por su buena precisión y velocidad en la detección de objetos. El sistema logró una alta efectividad en los experimentos con una persona invidente frente a problemas reales al detectar e informar sobre autos, buses y bancos próximos con precisiones de 96.95%, 87.5% y 55.5% respectivamente.

En el trabajo de (Huaroto Sevilla, 2019) se diseñó un Generador de Estímulos Mecánicos (GEM) económico y de simple manufactura que permita la retroalimentación kinestésica y táctil con una prótesis para la población con amputación a nivel transradial utilizando cuatro actuadores neumático blandos fabricados mediante

manufactura aditiva y moldeo. El esquema de funcionamiento del sistema diseñado se muestra en la figura 1.5. La validación de la investigación se realizó mediante sesiones con diez personas sanas y una persona con amputación. Los voluntarios reportaron sensaciones aprendidas de extensión y flexión de la muñeca, demostrando que la presión del sistema sobre la superficie del antebrazo está vinculada a la apertura y cierre de la mano.

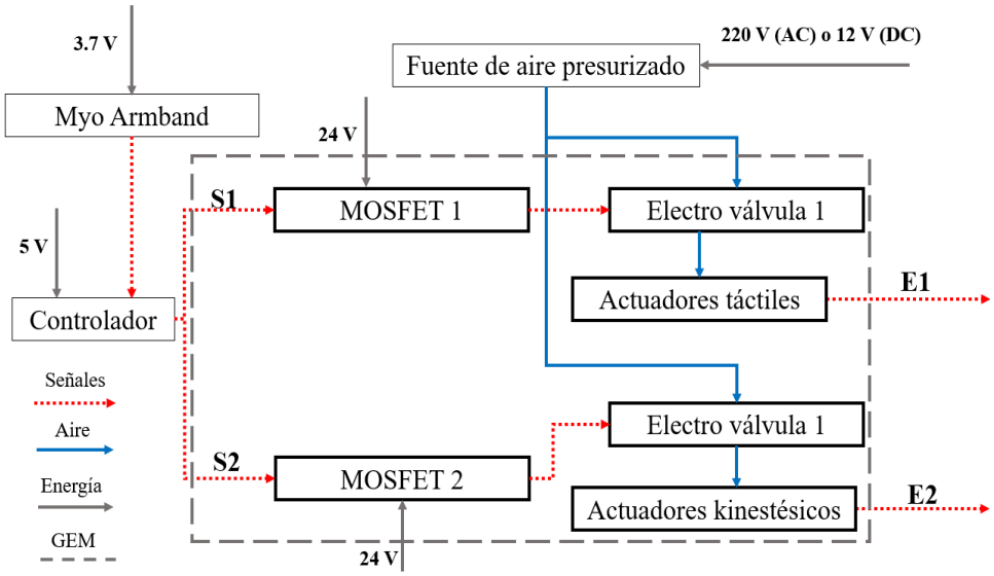


Figura 1.5. Esquema general de funcionamiento del sistema.

Fuente: Huaroto Sevilla. (2019) Esquema general de funcionamiento del sistema

CAPÍTULO II MARCO TEÓRICO Y CONCEPTUAL

Para el desarrollo de la tesis se ha conceptualizado términos asociados al movimiento de la mano; la electromiografía; procesamiento y clasificación de señales musculares; Interfaz Gráfica de Usuario; prótesis de mano y destreza, partiendo de una explicación general hasta conceptos específicos que ayudarán a un mejor entendimiento del presente estudio.

2.1. Marco teórico

2.1.1. Movimientos manuales

La mano humana y el cerebro son partes complementarias para cumplir las funciones de exploración y remodelación del mundo físico a través de la manipulación (Flanagan & Johansson, 2002). Los movimientos manuales pueden servir como “ventanas” por la cuales es posible aprender acerca de la representación subyacente de los objetos en la memoria y los procesos para los cuales se derivan y utilizan dichas representaciones (Lederman & Klatzky, 1987).

Los movimientos son descritos por diversos términos, y pueden considerarse también en pares opuestos (Moore et al., 2013). A continuación, se detalla los movimientos manuales principales:

- **Extensión y flexión**

La extensión significa un movimiento en una dirección posterior para acrecentar el ángulo entre los partes de la mano; la flexión, por otra parte, significa un movimiento en una dirección anterior para reducir el ángulo entre las partes de la mano, tal como se observa en la figura 2.1 (Moore et al., 2013).

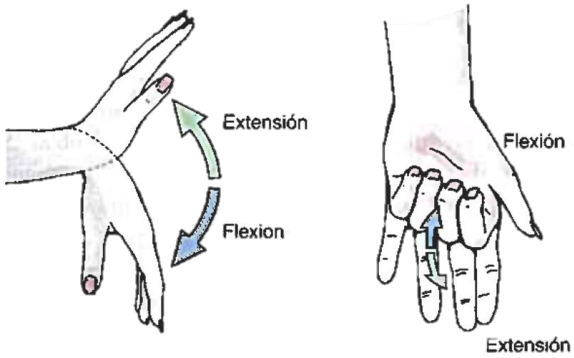


Figura 2.1. Extensión y flexión de la mano y de los dedos.
Fuente: Moore et al. (2013)

- **Oposición y Reposición**

La oposición realiza el contacto del pulgar con otro dedo, permitiendo realizar acciones como pellizcar, abotonar, sostener una taza; por otro lado, la reposición es el movimiento desde la oposición hasta la posición anatómica, tal como se observa en la figura 2.2 (Moore et al., 2013).

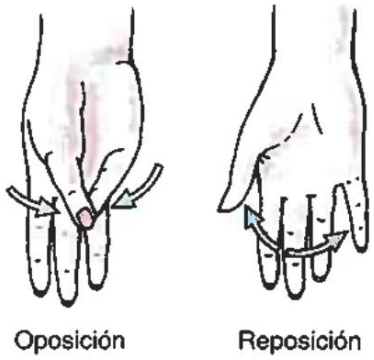


Figura 2.2. Oposición y reposición del pulgar.
Fuente: Moore et al. (2013)

2.1.2. Músculos dedicados al movimiento manual

Los músculos esqueléticos permiten el movimiento de la mano. Este tipo de músculo generalmente está unido a un hueso por los dos extremos mediante tendones (ver figura 2.3). El movimiento de los huesos, y por lo tanto de la mano, se produce cuando se contrae el tendón unido al músculo esquelético (Riaño & Quintero, 2010).

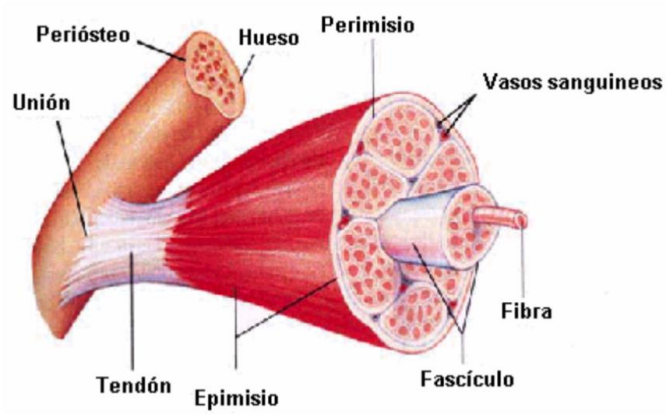


Figura 2.3. Músculo esquelético.
Fuente: Riaño & Quintero. (2010) Músculo esquelético

Las señales que se producen por los movimientos de los dedos se pueden captar en algunos músculos del antebrazo. Entre ellos destacan el músculo extensor común de los dedos, el músculo extensor largo del pulgar, el músculo flexor largo del pulgar, el músculo flexor común superficial de los dedos, el músculo palmar largo y el músculo extensor propio del meñique (Alzate Arias, 2018).

- **Músculo extensor común de los dedos**

Es el músculo extensor común de los dedos mediales: índice, medio, anular y meñique. Se origina en el húmero y forma un tendón por cada dedo medial, tal como se observa en la figura 2.4 (Drake et al., 2020).

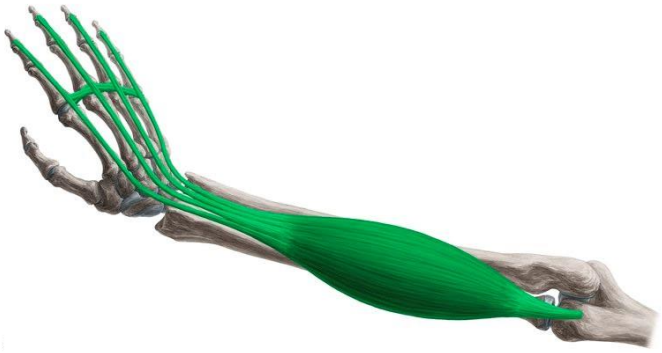


Figura 2.4. Músculo extensor común de los dedos.
Fuente: Kenhub.com (2022) Extensor digitorum muscle

- **Músculo extensor largo del pulgar**

El músculo extensor largo del dedo pulgar tiene como función extender todas las articulaciones del dedo que lleva su nombre. Se origina en el cúbito y en la membrana interósea, y llega hasta la superficie dorsal de la falange distal del dedo, tal como se muestra en la figura 2.5 (Drake et al., 2020).

Figura 2.5. Músculo extensor largo del pulgar.
Fuente: Kenhub.com (2022) Extensor pollicis longus muscle

- **Músculo flexor largo del pulgar**

El músculo flexor largo del dedo pulgar tiene como función flexionar el dedo que lleva su nombre. Se origina en el radio y en la membrana interósea, pasa a través del dedo pulgar y finalmente llega hasta el cimiento de la falange distal del pulgar, tal como se observa en la figura 2.6 (Drake et al., 2020).

Figura 2.6. Músculo flexor largo del pulgar.
Fuente: Kenhub.com (2022) Flexor pollicis longus muscle

- **Músculo flexor superficial de los dedos**

El músculo flexor superficial de los dedos tiene como función flexionar articulaciones de los dedos mediales: índice, medio, anular y meñique; asimismo, flexiona la articulación de la muñeca. Se origina en la cabeza humerocubital y en la cabeza radial del húmero y forma un tendón por cada dedo medial, tal como se observa en la figura 2.7 (Drake et al., 2020).

Figura 2.7. Músculo flexor superficial de los dedos.
Fuente: Kenhub.com (2022) Flexor digitorum superficialis muscle

- **Músculo palmar largo**

El músculo palmar largo es un accesorio de la muñeca y tiene como función oponerse a las fuerzas de desplazamiento de la epidermis de la palma cuando se produce un agarre. Este músculo tiene la particularidad de que se encuentra ausente en el 15% de la población (Drake et al., 2020). En la figura 2.8 se observa el músculo palmar largo.

Figura 2.8. Músculo palmar largo.
Fuente: Kenhub.com (2022) Palmaris longus muscle

- **Músculo extensor propio del meñique**

El músculo extensor del dedo meñique se considera un accesorio del dedo que lleva su nombre. Se origina en el húmero insertándose en la zona dorsal del dedo meñique tal como se observa en la figura 2.9 (Drake et al., 2020).

Figura 2.9. Músculo extensor propio del dedo meñique.
Fuente: Kenhub.com (2022) Extensor digiti minimi muscle

2.1.3. Electromiografía

Los cambios funcionales en el estado de las fibras musculares producen señales fisiológicas denominadas señales mioeléctricas; las cuales, mediante el método experimental electromiografía (EMG), son adquiridas, registradas y analizadas (Konrad, 2006).

La señal EMG se origina durante la activación de unidades motoras (MU). Una MU está conformada por una motoneurona y las fibras que esta activada mediante la transmisión de corriente eléctrica. En las neuronas ubicadas en la médula espinal, se produce un impulso nervioso llamado potencial de acción, este viaja mediante las

interneuronas hasta una motoneurona de un músculo esquelético y se propaga por sus fibras activando las células musculares para iniciar la contracción del músculo. La membrana de las células musculares cambia sus propiedades eléctricas y puede ser medido a través de electrodos EMG al sumar los potenciales de acción de las unidades motoras (MUAP) como se visualiza en la figura 2.10 (Pequera, 2015).

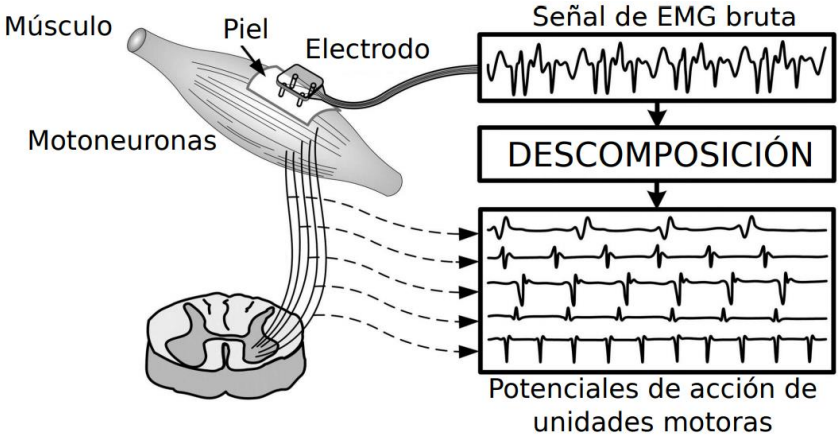


Figura 2.10. Esquema del registro EMG y la descomposición de 5 MUAP.
Fuente: Pequera. (2015)

La electromiografía puede ser un factor importante en el diagnóstico de afecciones que afectan los nervios y las fibras musculares esqueléticas. El propósito de la prueba no es hacer un diagnóstico de una enfermedad, sino evaluar el estado funcional de los nervios y las fibras del músculo esquelético (Perotto, 2011).

2.1.4. Electrodos EMG

Los electrodos EMG son conductores eléctricos utilizados en la EMG para medir la actividad bioeléctrica de los músculos. Existen 2 tipos principales de electrodos EMG: superficiales e insertados. Los electrodos insertados tienen otros dos tipos: electrodos de aguja y de alambre fino (Naik, 2012).

2.1.5. Electrodos EMG superficiales

Los electrodos EMG superficiales proporcionan un procedimiento no invasivo para mensurar la señal EMG. A diferencia de los electrodos de insertados, los electrodos superficiales, son simples de implementar y no requiere una estricta supervisión y certificación médica. Los electrodos superficiales se está utilizando con mayor frecuencia para controlar prótesis para la población con discapacidad física y amputados (Naik, 2012).

Los dos tipos de electrodos superficiales comúnmente usados son: El electrodo EMG seco, el cual está en contacto con la piel; y el electrodo EMG gelificados, el cual utiliza un gel electrolítico entre la epidermis y el electrodo (Day, 2002).

- **Electrodos EMG seco**

Los electrodos secos no requieren gel para medir las señales EMG en la epidermis. Los electrodos de barra (ver figura 2.11) y los electrodos de matriz son ejemplos de electrodos secos. Estos electrodos pueden contener más de una superficie de detección y en muchos ejemplos, también se puede emplear un circuito de preamplificación interno (Naik, 2012).

Figura 2.11. Electrodo EMG seco de Barra Reutilizable.
Fuente: Naik. (2012) A Reusable Bar Dry EMG Electrode

- **Electrodos EMG gelificados o húmedos**

Los electrodos EMG gelificados o húmedos necesitan una sustancia electrolítica gelificada como interfaz entre la epidermis y los electrodos como se visualiza en la figura 2.12. El Ag-AgCl es la sustancia que se utiliza con mayor frecuencia como componente metálico de los electrodos húmedos. La capa AgCl habilita que la corriente muscular fluya con más facilidad en la epidermis, gel conductor y electrodo (Day, 2002).

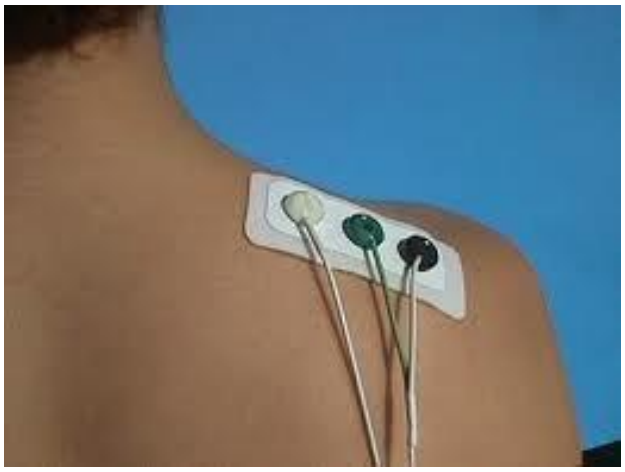


Figura 2.12. Electrodos EMG gelificado o húmedos.
Fuente: Naik. (2012) Gelled or Wet EMG Electrodes

2.1.6. *Windowing*

La técnica de *Windowing* es un proceso recurrente para la clasificación de actividades utilizando sensores fisiológicos instalados en las partes del cuerpo, además su esencia permite aplicaciones en tiempo real. La técnica consiste en dividir la señal del sensor en segmentos de tiempo cortos llamados ventanas, y secuencialmente aplicar técnicas de procesamiento y/o algoritmos de clasificación a cada ventana (Preece et al., 2009).

Existen tres técnicas de *Windowing*: Ventanas deslizantes, ventanas definidas por eventos y ventanas definidas por actividades. Siendo la técnica de ventana deslizante ideal para aplicaciones de tiempo real debido a que no requiere procesamiento previo de la señal del sensor. La técnica de ventanas deslizantes de una señal de sensor corporal continua al dividir en partes de longitud fija llamado ventanas se muestra en la figura 2.13 (Preece et al., 2009).

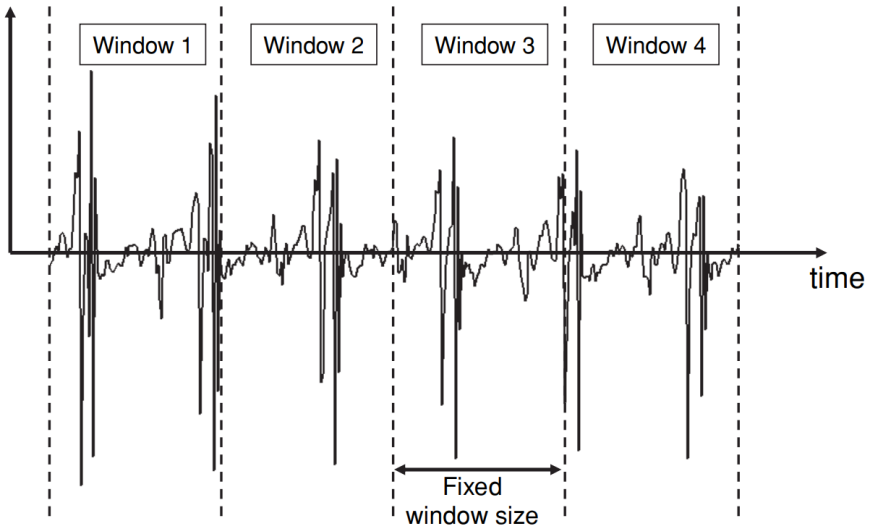


Figura 2.13. Ventanas deslizantes a lo largo de una señal continua.
Fuente: Preece et al., (2009) Sliding windows along a continuous signal

2.1.7. Estandarización

El libro “Estadística básica para ciencias de la salud” (Montanero & Minuesa, 2018) define la estandarización como un procedimiento estadístico aplicado a un muestra para disminuir a cada uno de sus datos dato X_i su promedio aritmético $\bar{x}$, para luego, ser dividido por su desviación típica σ, tal como se observa en la ecuación 2.1.

$$z_i = \frac{x_i - \bar{x}}{\sigma_x} \qquad (2.1)$$

La estandarización ajusta una muestra en un modelo de distribución denomina normal estándar el cual tiene un promedio de 0 y una desviación típica igual a 1 para que los datos en diferentes niveles tengan un nivel común. En general los valores más distantes al eje de distribución forman un 5% total de la cantidad de las datos, tal como se muestra en la figura 2.14 (Montanero & Minuesa, 2018).

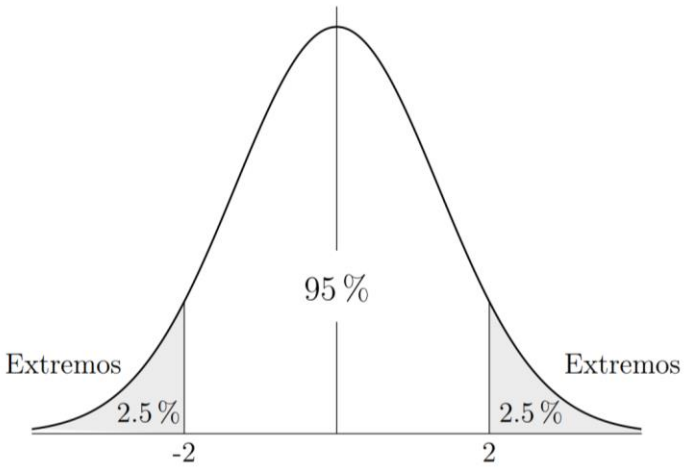


Figura 2.14. Distribución normal.
Fuente: Montanero & Minuesa (2018) Distribución normal

2.1.8. Filtros para señales EMG

Un filtro es un dispositivo que modifica una señal, también se nombrados como selectores de frecuencia debido a que permiten el paso de las señales en algunas frecuencias y restringen el paso en otras frecuencias (Miyera, 2004).

Los filtros digitales son más utilizados que los filtros analógicos para el procesamiento de señales biológicas. Los filtros analógicos tienen dos desventajas: no muestra la cantidad de interferencia en la señal que se va utilizar, y la señal no se puede regenerar con la posibilidad de perder información (Zschorlich, 1989).

Por otra parte, con un método de filtrado digital, se puede guardar hardware adicional y se puede configurar varios tipos de filtrado en poco tiempo con la programación. Otras ventajas, en comparación al filtro analógico, es la independencia de problemas con el hardware, tolerancias de producción, temperatura y cambios de características debido al envejecimiento (Zschorlich, 1989).

La electromiografía de superficie (EMG) normalmente se corrompe por tres tipos de ruidos: la interferencia de la línea eléctrica, el ruido gaussiano blanco y la desviación de la línea de base (X. Zhang & Zhou, 2013). Es por ello la necesidad de aplicar filtros digitales, capaces de eliminar o reducir estos ruidos.

Como se observa en la figura 2.15, los filtros se pueden distinguir entre la función del rango de frecuencia que dejen pasar o rechazar, los cuales son:

- **Filtro pasa bajo:** Es un filtro que deja pasar frecuencias desde los 0 Hz hasta cierto valor de frecuencia (Gallardo, 2019).
- **Filtro pasa banda:** Este filtro permite el paso en un conjunto continuo de frecuencias y elimina el resto (Gallardo, 2019).
- **Filtro pasa alto:** Este filtro pasa todas las frecuencias mayores a cierto valor de frecuencia (Gallardo, 2019).
- **Filtro rechaza banda:** Es un filtro que elimina un determinado rango de frecuencia, y deja pasar el resto (Gallardo, 2019).

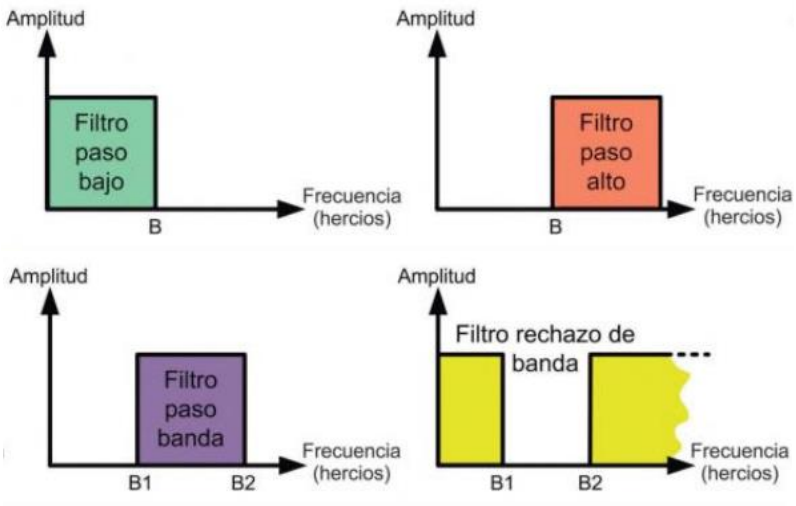


Figura 2.15. Tipos de filtros.
Fuente: Gallardo (2019) Tipos de filtros

2.1.9. Transformada Wavelet

La transformada Wavelet (WT) es un mapeo de una señal de tiempo a la representación conjunta de escala de tiempo, utilizado como herramienta analizadora de señales de banda ancha transitorias rápidas y no estacionarias (Dorf et al., 2018).

La WT es potente para el análisis de espectro local multiresolución de señales como de sonido, radar, sonar, sísmicas, fisiológicas; asimismo, para procesamiento de imágenes y reconocimiento de patrones (Dorf et al., 2018).

- **Transformada Wavelet Continua**

La Transformada Wavelet Continua (CWT) es un ejemplo de WT, el cual descompone una función $f(t)$ en un conjunto de funciones básicas $h_{s,\tau}(t)$ llamadas wavelets, teniendo en cuanta que $f(t)$ pertenece a un espacio vectorial de funciones medibles e integrables al cuadrado (Dorf et al., 2018). La ecuación 2.2 define la CWT de la función $f(t)$, donde * significa la conjugada compleja.

$$W_f(s,\tau) = \int f(t) h^*_{s,\tau}(t)\, dt \qquad (2.2)$$

La mayoría de wavelets son valores reales y son generadas desde una sola y básica función wavelet (wavelet madre) $h(t)$ con escalamiento y translación (Dorf et al., 2018), como se observa en la ecuación 2.3.

$$h_{s,\tau}(t) = \frac{1}{\sqrt{s}} h\left(\frac{t-\tau}{s}\right) \qquad (2.3)$$

Donde $\boldsymbol{s}$ es el valor de escalamiento y $\boldsymbol{\tau}$ es el valor de desplazamiento. Las $h_{s,\tau}(t)$ obtenidas desde una wavelet madre poseen diferentes escalas s y locaciones τ, sin embargo, todos mantienen la misma estructura (Dorf et al., 2018).

- **Escalograma**

Es una representación de tiempo y frecuencia (2-D) de una señal. En algunos casos, se forma a partir de la Transformada Wavelet Continua (CWT), utilizando varios parámetros de escala $\boldsymbol{s}$, desplazamiento $\boldsymbol{\tau}$, y función de wavelet madre (Nahid et al., 2020a).

Los escalogramas wavelet $W_f(s,\tau)$ comunican la propiedad de localización de la escala de tiempo (frecuencia) de la CWT de una señal $f(t)$ (Belkhou et al., 2017).

Un ejemplo es el trabajo de (Shin et al., 2022), donde se crearon escalogramas a partir de las señales electrocardiográficas (ECG) para investigar sus características de frecuencia, tal como se muestra en la figura 2.16.

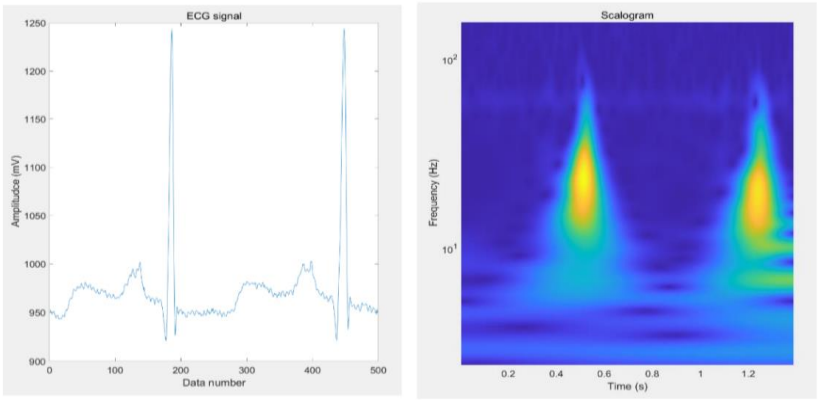


Figura 2.16. Una señal de ECG y su escalograma.
Fuente: Shin et al. (2022) An ECG signal and its scalogram

2.1.10. Deep Learning

El Deep Learning es una técnica Machine Learning que explora numerosas capas de información procesada con el fin de extraer y transformar características; asimismo, analizar y estimar patrones de datos, señales, imágenes, etc. (Deng & Yu, 2014).

El Deep Learning es una perspectiva de la Inteligencia Artificial (AI) que aprende a simbolizar el mundo como un sistema anidado de conceptos, de manera que cada concepto se define en relación con otros más sencillos, a su vez, simbolizaciones con mayor profundidad y abstracción computacional que otras (Goodfellow et al., 2016).

En el control de dispositivos médicos para la ayuda y rehabilitación de pacientes, la Inteligencia Artificial y el Deep Learning ha tenido un enfoque vanguardista. Por ejemplo, con el objetivo de mejorar la funcionalidad de las prótesis de mano, trabajos de investigaciones ha aplicado la estimación de patrones de movimientos manuales a través de la adquisición y clasificación de señales EMG (Sarmiento, 2020).

2.1.11. Red Neuronal Convolucional

Entre diferentes arquitecturas del Deep Learning, resalta una Red Neuronal de varias capas denominada Red Neuronal Convolucional (CNN). Esta Red posee una arquitectura que permite una mayor disposición para generalizar que las redes de capas totalmente conectadas *(FC Layers*), resultado en un aprendizaje e identificación más eficiente de características con grados altos de abstracción, en especial los datos espaciales (Ghosh et al., 2020).

Un número finito de capas de procesamiento de información componen un modelo de Red Neuronal Convolucional. Estas capas permiten un aprendizaje a partir de las diferentes características de la información entrante a diferentes niveles de abstracción. Las capas introductorias extraen las características de menor abstracción, mientras que las capas con mayor profundidad extraen las características de mayor abstracción (Ghosh et al., 2020). En la figura 2.17 se muestra el modelo básico de la Red Neuronal Convolucional, en el que resalta las capas convoluciones; las capas de *Pooling* (reducción de dimensiones), y la capa totalmente conectada.

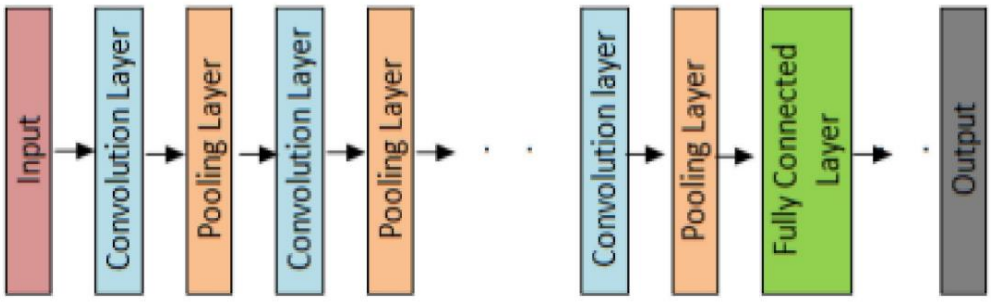


Figura 2.17. Modelo Conceptual de la Red Neuronal Convolucional.
Fuente: Ghosh, et al. (2020) Modelo Conceptual de la Red Neuronal Convolucional

2.1.12. Interfaz Gráfica de Usuario

La Interfaz Gráfica de Usuario (GUI) es medio por el cual las personas y las computadoras se comunican e interactúan entre sí, para resolver el problema *"Blank screen"*, que surge cuando la computadora no realiza ninguna indicación que un usuario debería hacer a continuación (Jansen, 1998). Las GUI toman ventaja de las capacidades gráficas de las computadoras para facilitar la comunicación al ocultar los detalles del lenguaje de programación del usuario utilizando *widgets* como ventanas, íconos , botones, gráficas, cuadros de diálogo, entre otros, que se activan cuando el usuario los manipula con un mouse u otro dispositivo señalador (Martinez, 2011).

- **Principios de diseño de una GUI**

En el trabajo de (Martinez, 2011) se definieron 4 principios que para realizar un correcto diseño de una GUI. A continuación, se definen los principios:

- Principio 1: 'Enfócate en los usuarios y sus tareas, no en la tecnología'. Significa que el diseñador debe comenzar el proceso del desarrollo de la GUI teniendo en cuenta: las personas que lo utilizarán, las actividades que debe cumplir y los problemas que debe resolver.
- Principio 2: 'Considerar la función primero, la presentación después'. El desarrollador debe considerar el propósito de la GUI antes de cualquier codificación.

- Principio 3: 'Conforme la visión del usuario, y no la complique más'. La idea general de crear una GUI debe ser hacer las cosas más fáciles para el usuario, no hacerlas más complejas.
- Principio 4: 'Promover el aprendizaje y entregar información, no solo datos'. El diseño de la GUI debe enforcarse en la facilidad de uso y de aprendizaje; por ejemplo, agrupando controladores por funcionalidad y brindando información útil como títulos, etiquetas, etc.

- **GUI para la visualización de señales fisiológicas**

Una tarea común en la investigación biomédica es registrar y visualizar señales fisiológicas (electromiografía, electrocardiografía, etc.) en tiempo real. Comúnmente, las GUI de este tipo de señales se basan en herramientas asociadas con un proveedor de dispositivos de adquisición de señales; sin embargo, existe la posibilidad de diseñar una GUI bajo paquetes de software de código abierto para poder visualizar, registrar y procesar señales fisiológicas. Un ejemplo destacable es RTGraph (bajo licencia MIT), debido a que es una Interfaz Gráfica de código abierto con el objetivo de mostrar señales en tiempo real y exportarlas a un archivo “CSV” (Sepúlveda et al., 2015).

En la figura 2.18 se muestra una GUI utilizando RTGraph para mostrar 3 gráficas: Una señal EMG, una estimación de fatiga y tres señales de aceleración (Sepúlveda et al., 2015).

Figura 2.18. Captura de pantalla de una GUI RTGraph personalizada.
Fuente: Sepúlveda et al. (2015) Screenshot of a RTGraph GUI customized

2.1.13. Prótesis de mano

El desarrollo y fabricación de las prótesis de mano ha sido gracias al avance tecnológico y la biomecánica del ser humano. El objetivo de una prótesis es proporcionar funciones que se han perdido por un miembro faltante para restablecer la calidad de la vida de la población que padece de alguna amputación. La elección de una prótesis está condicionada a agentes como: el nivel de amputación, la funcionalidad y el factor económico. Es por ello que las prótesis se han desarrollado con diferentes niveles de tecnologías (Brito Guaricela et al., 2013).

- **Prótesis de mano mioeléctrica**

La prótesis de mano mioeléctrica está basado en la captura de señales EMG generadas en los músculos esqueléticos para reflejar la

intención del usuario utilizando esquemas de descifrado como el reconocimiento de patrones. Las investigaciones del descifrado de la información de las señales EMG son recurrentes debido a que significa mejorar la destreza de la prótesis de mano (Geethanjali, 2016). En la figura 2.19 se muestra los componentes típicos de las prótesis transradial con control mioeléctrico, dónde se destacar el uso de motores, microcontroladores y electrodos. En este tipo de prótesis, normalmente se utiliza sensores EMG superficiales no invasivos debido a sus ventajas en la comodidad y en la facilidad de uso para la mayoría de amputados en comparación de los electrodos invasivos (Calado et al., 2019).

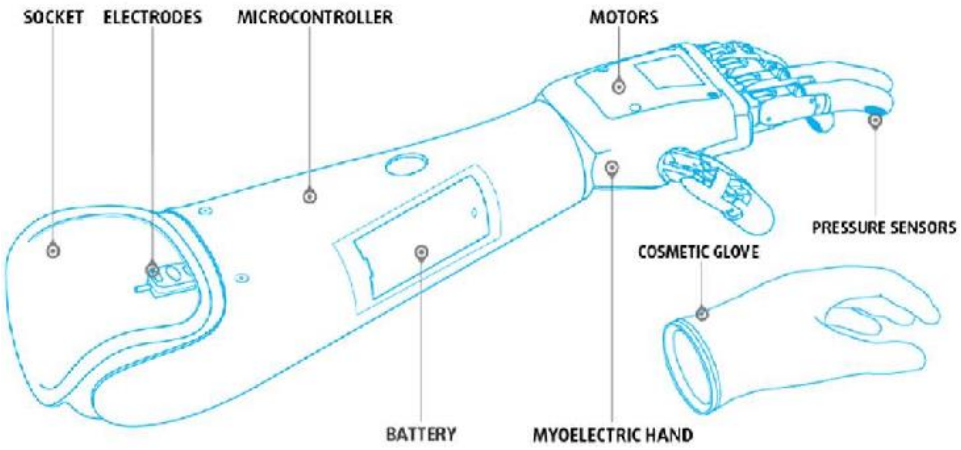


Figura 2.19. Componentes típicos de una prótesis mioeléctrica.
Fuente: Calado et al. (2019) Typical components of a myoelectric prothesis

- **Prótesis de mano de agarre simple**

Las prótesis de agarre simple son manos artificiales de un grado de libertad para sostener un objeto. El usuario puede controlar la postura de los dedos mediante la contracción o relajación de un músculo o grupo de músculos (Chappell et al., 1987).

Las prótesis de mano de agarre simple normalmente tienen un tipo de control mioeléctrico de encendido/apagado (velocidad constante) o un esquema control de velocidad proporcional al nivel de intensidad de las señales EMG (Geethanjali, 2016). En ambos casos, el control solamente logra abrir y cerrar los dedos de la prótesis, lo cual limita al usuario a agarrar objetivos como papeles, lapiceros, etc.

2.1.14. Convección Denavit - Hartenberg

La convección (Denavit & Hartenberg, 1955) es una descripción de las relaciones cinemáticas entre los enlaces conectados por uniones como rotaciones o deslizamientos de un grado de libertad. En la figura 2.20 se observa los 4 parámetros que utiliza la convección DH en cada articulación de eslabones los cuales pueden ser identificados siguiendo un procedimiento determinado (Rocha et al., 2011).

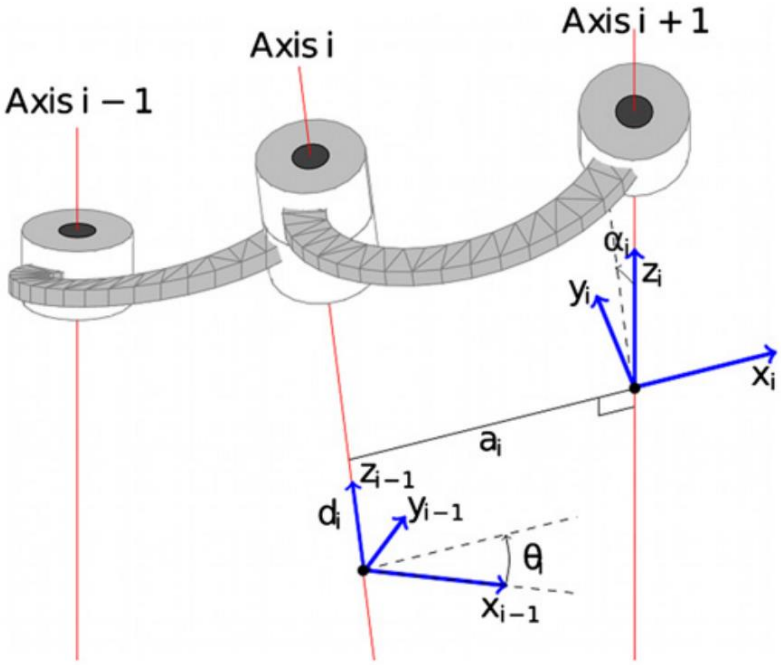


Figura 2.20. Parámetros Denavit - Hartenberg.
Fuente: Rocha et al. (2011) Denavit-Hartenberg parameters

Mediante la convección D-H, en cada articulación se determina una matriz homogénea $A_i^{i-1}(q_i)$ el cual está compuesto por rotaciones y traslaciones de los valores de los parámetros D-H como se muestra en la ecuación 2.4 (Rocha et al., 2011).

$$A_i^{i-1}(q_i) = Rot_{(z_{i-1},\theta_i)} Trans_{(z_{i-1},d_i)} Trans_{(x_i,a_i)} Rot_{(x_i,\alpha_i)} \quad (2.4)$$

El desarrollo de la ecuación 2.4 se observa en la ecuación 2.5, dónde C y S son referidos a las razones trigonométricas coseno y seno respectivamente (Rocha et al., 2011).

$$A_i^{i-1}(q_i) = \begin{bmatrix} \mathrm{C}_{\theta_i} & -\mathrm{S}_{\theta_i}\mathrm{C}_{\alpha_i} & \mathrm{S}_{\theta_i}\mathrm{S}_{\alpha_i} & a_i\mathrm{C}_{\theta_i} \\ \mathrm{S}_{\theta_i} & \mathrm{C}_{\theta_i}\mathrm{C}_{\alpha_i} & -\mathrm{C}_{\theta_i}\mathrm{S}_{\alpha_i} & a_i\mathrm{C}_{\theta_i} \\ 0 & \mathrm{S}_{\alpha_i} & \mathrm{C}_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

La matriz homogénea se divide en una matriz de rotación $R_i^{i-1}(q_i)$ y un vector de posición del efecto final $P_i^{i-1}(q_i)$ cómo se observa en la ecuación 2.6 (Rocha et al., 2011).

$$R_i^{i-1}(q_i) = \begin{bmatrix} \mathrm{C}_{\theta_i} & -\mathrm{S}_{\theta_i}\mathrm{C}_{\alpha_i} & \mathrm{S}_{\theta_i}\mathrm{S}_{\alpha_i} \\ \mathrm{S}_{\theta_i} & \mathrm{C}_{\theta_i}\mathrm{C}_{\alpha_i} & -\mathrm{C}_{\theta_i}\mathrm{S}_{\alpha_i} \\ 0 & \mathrm{S}_{\alpha_i} & \mathrm{C}_{\alpha_i} \end{bmatrix} \wedge\ P_i^{i-1}(q_i) = \begin{bmatrix} a_i\mathrm{C}_{\theta_i} \\ a_i\mathrm{C}_{\theta_i} \\ d_i \end{bmatrix} \quad (2.6)$$

2.1.15. Destreza

Según (Gonzalez et al., 2015). la destreza, para las prótesis de mano, puede definirse como la habilidad de los dedos y la mano para moverse con un fin. Es una habilidad muy valiosa y versátil que se puede tomar como una variable que se relaciona con la competencia de realizar actividades de la vida diaria (AVD). Por lo tanto, las prótesis de alta destreza estén facultadas a clasificar movimientos manuales que permitan diferentes tipos de agarres.

Según (Dollar, 2014), a partir de la taxonomía de sujeción de Cutkosky (ver figura 2.21) se puede usar para crear un conjunto de tareas estándar con el objetivo de asignar un puntaje general, comparar y cuantizar la destreza manual de un robot como las prótesis.

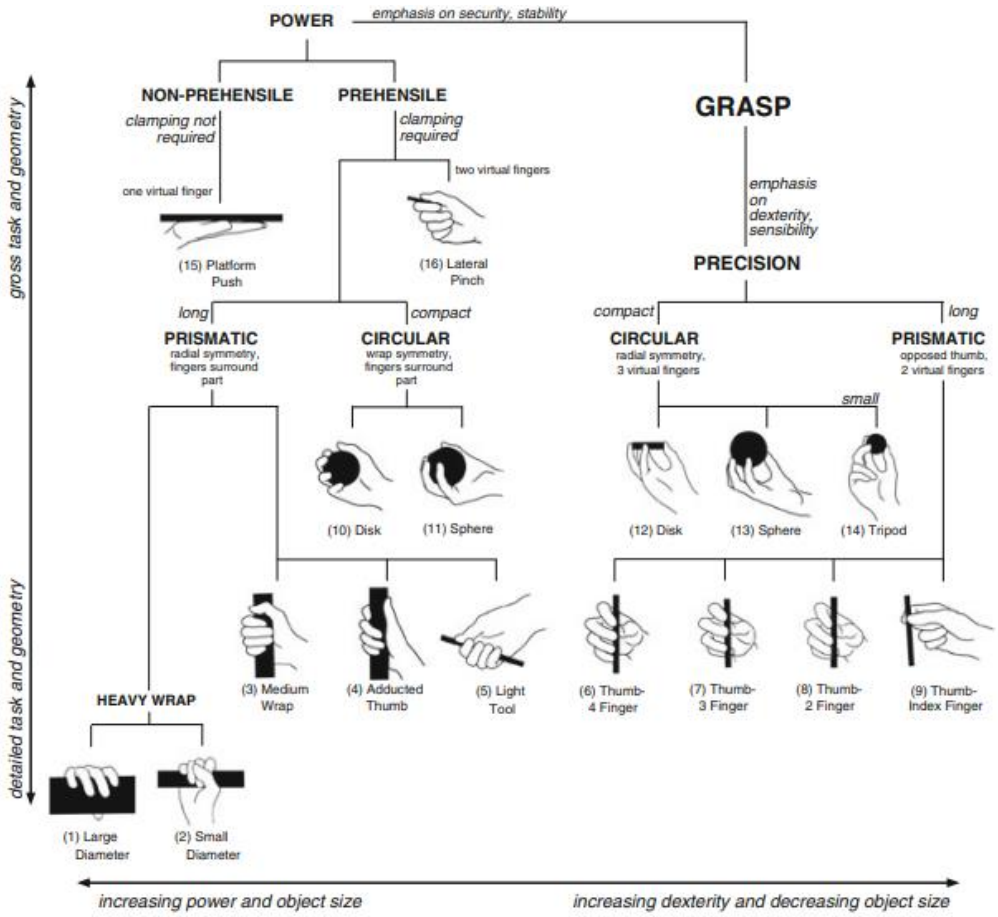


Figura 2.21. Taxonomía de sujeción de Cutkosky.
Fuente: Dollar, et al. (2014) Cutkosky grasp taxonom

Por otro lado, en el estudio de (Cheu et al., 2005) se planteó un índice para valorar cuantitativamente la destreza de una prótesis. Los dos principales atributos considerados fueron el desempeño del agarre y la manipulación. Los valores de los parámetros se muestran en la figura 2.22, y la destreza se calcula con la ecuación 2.7.

ATTRIBUTE DEXTERITY.

	Abilities	Characteristics		Val
Grasping	Grasping with the tip of the fingers. Precision W_{G1}=0.6	Number of fingers G_{C1}=0,2	2 fingers	0,5
			3 fingers	1
		Degrees of freedom G_{C2}=0,4	3 Degrees	1
			< 3 degrees	0,5
		Forces of it grabs G_{C3}=0,4	Similar 15 N	1
			< 15 N	0,5
	Grasping performance W_{G2}=0.4	Number of fingers G_{C1}=0,2	2 fingers	0,5
			3 fingers	1
		Degrees of freedom G_{C2}=0,3	3 Degrees	1
			< 3 degrees	0,5
		Forces of it grabs G_{C3}=0,4	Similar 15 N	1
			< 15 N	0,5
		Palm G_{C4}=0,1	Palm existence	1
			Non existence	0
Manipulation	Manipulation with the tip of the fingers. Precision W_{M1}=0.6	Number of fingers M_{C1}=0,2	2 fingers	0,5
			3 fingers	1
		Degrees of freedom M_{C2}=0,4	10 degrees	1
			Between 3 and 9	0,15-0,85
		Forces of it grabs M_{C3}=0,4	Similar 15 N	1
			< 15 N	0,5
	Movement of the wrist W_{M2}=0.4	Rotation		0,5
		Adduction/abduction		0,25
		Flexor and extensor		0,75
		Combination of the previous ones		1

Figura 2.22. Tabla de atributos de destreza.
Fuente: Cheu et al. (2005) Table Attribute Dexterity

$$Destreza = \sum_{i=1}^{4}\sum_{j=1}^{2}\left(A_{Gj} * Val_i\right) + \sum_{i=1}^{2}\left(A_{Mi} * Val_i\right) \qquad (2.7)$$

Donde $A_{Gj}(= W_{Gj} * G_{Cj})$ y $A_{Mi}(= W_{Mi} * M_{Ci})$ corresponde a los diferentes pesos de agarre y manipulación de las habilidades respectivamente, y Val_i es la característica de la prótesis.

Tanto en el trabajo (Dollar, 2014) como de (Cheu et al., 2005), se concluye que la destreza manual es proporcional a la cantidad de movimientos manuales de agarre que una prótesis puede realizar con relación a las AVD.

2.1.16. Clasificación de movimientos manuales de agarre

En el trabajo de (Vergara et al., 2012) se realizó una clasificación de nueve movimientos manuales de agarre del trabajo a partir de 180 videos de personas realizando AVD. En la Tabla 2.1 se observa la taxonomía de la clasificación de agarres, mientras que en la figura 2.23 se observan los agarres utilizados en actividades cotidianas.

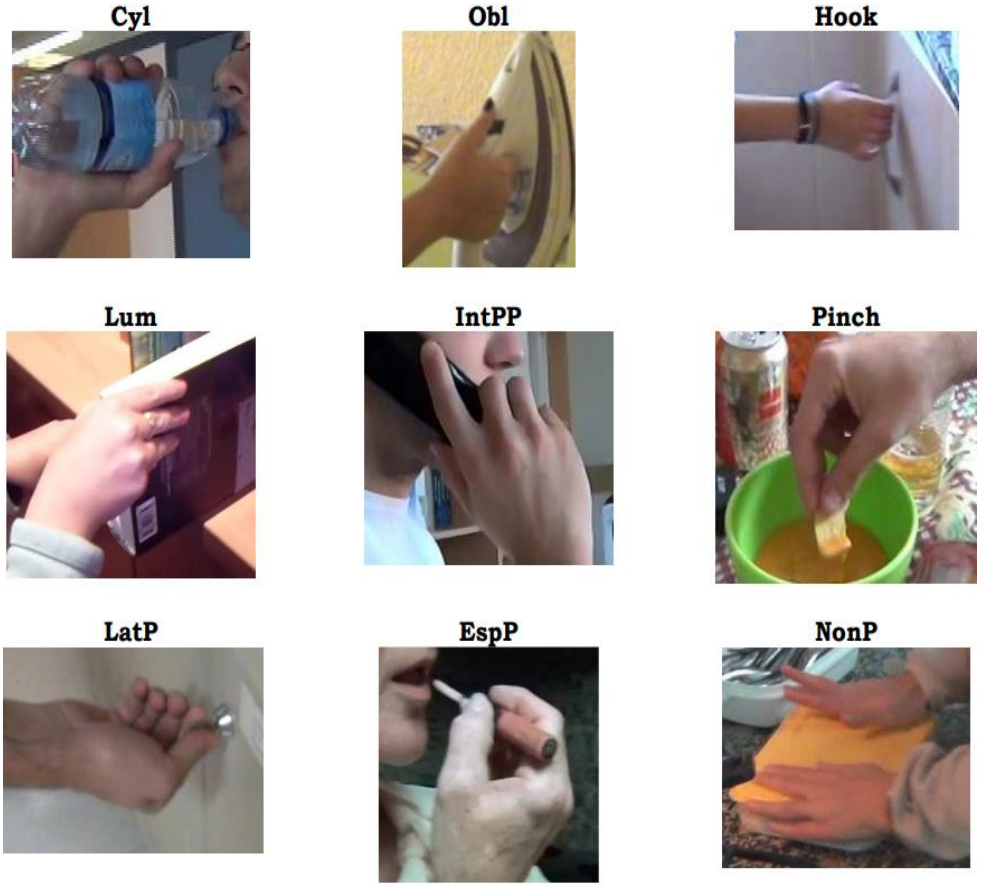


Figura 2.23. Clasificación de agarres.
Fuente: Vergara et al. (2012) Clasificación de agarres

Tabla 2.1. Taxonomía de la clasificación de agarres.

Agarre	Taxonomía
Agarre cilíndrico	Cyl
Agarre palmar oblicuo	Obl
Agarre de gancho	Hook
Agarre lumbrical	Lum
Agarre intermedio de potencia-precisión	IntPP
Pinza	LatP
Pinza lateral	Pinch
Pinza especial	EspP
Movimiento no prensil	NonP

Fuente: Elaboración propia en base del trabajo de Vergara et al. (2012)

2.2. Marco Conceptual

2.2.1. Movimientos manuales

Los movimientos manuales son medios por el cual una persona puede explorar, remodelar, y aprender sobre el mundo físico. Los movimientos manuales más utilizados son la extensión y flexión de los dedos los cuales permiten sostener objetos de distintas formas.

2.2.2. Electromiografía

Es una técnica por el cual se puede adquirir señales mioeléctricas (musculares) mediante electrodos que capturan la suma de MUAP en la membrana del músculo. Las señales captadas por los electrodos deben ser amplificadas y pasar por un procesamiento de filtros para ser analizadas.

2.2.3. Escalograma

Es una imagen que representa el tiempo y frecuencia de una señal al ser construida por la Transformada Wavelet Continua (CWT). Contiene información importante como la localización de la frecuencia

a lo largo del tiempo. Los investigadores han venido empleando el escalograma de las señales biológicas como imágenes de entradas de modelos de Deep Learning para clasificar patrones.

2.2.4. Deep Learning

El Deep Learning es una parte del Machine Learning que busca imitar la arquitectura del cerebro humano mediante Redes Neuronales Artificiales. Este enfoque de la AI ha logrado grandes desempeños en reconocimiento de voz, detección de objetos, clasificación de imágenes, etc. En el área de la biomedicina, el Deep Learning se ha vuelto un instrumento primordial en el reconocimiento de patrones.

2.2.5. Destreza que ofrece el sistema de control

La destreza para las prótesis es una capacidad que ayuda a realizar activades de la vida diaria (AVD). En el trabajo de (Dollar, 2014) se menciona que a partir de la taxonomía de sujeción de Cutkosky (ver figura 5.21) se puede crear estándares para comparar la destreza manual de las prótesis. Por consiguiente, para la presente investigación, se plantea un estándar (ver tabla 2.2) que compare el grado de destreza que ofrece el sistema de control en base a la clasificación de los movimientos de taxonomía de sujeción de Cutkosky y la de tipos de agarre de (Vergara et al., 2012). Por lo que se asignará un puntaje según la precisión de la clasificación en tiempo real de los movimientos (P_i) con un coeficiente proporcional al grado de destreza del movimiento según la taxonomía de sujeción de Cutkosky (C_i), tal como se observa en la ecuación 2.8.

Tabla 2.2. Estándar de la destreza que ofrece el sistema de control.

Taxonomía según Cutkosky				**Taxonomía según (Vergara et al., 2012)**	**C_i (Si clasifica al menos uno)**
Poder (enfatiza la seguridad y estabilidad)	No prensil (no se requiere sujeción)		*Platform Push* (Empuje de plataforma)	Movimiento no prensil (NonP)	$C_1 = 0.1$
	Prensil (se requiere precisión)	Prismático (simetría radial) o circular (simetría de envoltura)	*Large Diameter* (diámetro grande)	Agarre cilíndrico (Cyl)	$C_2 = 0.15$
			Small Diameter (diámetro pequeño)	-	
			Medium Wrap	-	
			Adducted Thumb (pulgar aducido)	Agarre palmar oblicuo (Obl)	
			Light tool (herramienta ligera)	-	
			Disk (disco con palma)	-	
			Sphere (esfera con palma)	-	
		Con dos dedos	*Lateral Pinch* (Pinza lateral)	Pinza lateral (LatP)	$C_3 = 0.2$
Precisión (enfatiza la destreza y sensibilidad)	Circular (simétrico con agarre de 3 dedos)		*Disk* (disco sin palma)	-	$C_4 = 0.25$
			Sphere (esfera sin palma)	-	
			Tripod (trípode)	-	
	Prismático (pulgar opuesto)		*Thumb- 4 Finger* (agarre pulgar y 4 dedos)	Agarre lumbrical (Lum)	$C_5 = 0.3$
			Thumb- 3 Finger (agarre pulgar y 3 dedos)	-	
			Thumb- 2 Finger (agarre pulgar y 2 dedos)	-	
			Thumb- 1 Finger (agarre pulgar e índice)	Pinza (Pinch)	

Fuente: Elaboración propia en base del trabajo de (Dollar, 2014) y Cheu et al. (2005)

$$\Delta Destreza = \sum_{i=1}^{5} (C_i * P_i) \qquad (2.8)$$

2.2.6. Prótesis de mano

La prótesis de mano tiene como propósito permitir a una persona con amputación o discapacidad motora en la mano, restablecer algunas funciones para permitirle realizar todas, o parcialmente las actividades de la vida diaria (AVD). El tipo más destacado es la prótesis de mano mioeléctrica debido a su naturaleza de adquirir, procesar y clasificar señales musculares la cual permite la clasificación de movimientos manuales según la intención del usuario.

En ejemplo de prótesis de mano son las prótesis de agarre simple las cuales tienen un control de un solo grado de libertad para abrir y cerrar los dedos. Ello limita a los usuarios a no poder agarrar objetivos pequeños como monedas u objetos planos como hojas o tarjetas; por lo tanto, su grado de destreza es bajo.

CAPÍTULO III DESARROLLO DE LA TESIS

3.1. Metodología del desarrollo de la tesis

El trabajo de investigación de tesis tiene como fin principal el diseño de un sistema de control mioeléctrico con clasificación en tiempo real de movimientos manuales basado en Deep Learning para mejorar la destreza de una prótesis de mano de agarre simple, para ello se propone una metodología cimentada en la integración de 3 subsistemas (adquisición, procesamiento y clasificación) y una Interfaz Gráfica de Usuario (GUI) de señales EMG. Posteriormente, para validar el sistema se implementa una simulación de una prótesis de mano conectado a la GUI mediante el cual los voluntarios realizaran pruebas de destreza. En la figura 3.1 se observa el flujograma del desarrollo de la tesis.

En el subsistema de adquisición de señales EMG se seleccionará el tipo y cantidad de sensores EMG; asimismo, el posicionamiento de los electrodos en los músculos del antebrazo. Posteriormente, se realizará la digitalización de las señales EMG a través la tarjeta Raspberry Pi Pico el cual, mediante comunicación serial, permitirá adquirir y visualizar las señales EMG cuando se realice diferentes movimientos manuales.

A partir de las señales EMG adquiridas, se obtendrá imágenes que representan el tiempo y frecuencia de señales (escalogramas) a través del subsistema de procesamiento. El diseño del subsistema empieza con la técnica de "Windowing", el cual permite la división de las señales en lapsos de tiempos iguales llamado ventanas. Cada ventana pasará por la técnica de la estandarización para centrar las señales y uniformizar el rango de valores de los sensores. Posteriormente, las ventanas serán filtradas mediante el filtro pasabanda y el filtro notch para atenuar ruidos de compensación de corriente continua, ruidos generados por la red eléctrica, y otros. Finalmente, las ventanas serán transformadas en escalogramas mediante la Transformada Wavelet Continua (CWT).

El siguiente paso es el diseño del subsistema de clasificación de movimientos manuales a partir del entrenamiento de escalogramas con 4 arquitecturas de Deep Learning denominadas Redes Neuronales Convolucionales (CNN), los cuales son: *AlexNet*, *ResNet*, *DenseNet* y *MobileNet V2*. El objetivo del subsistema es poder obtener un modelo de clasificación con alta precisión y baja pérdida de entropía cruzada (CEL), para ello se realizará una comparación del entrenamiento de 20 modelos por cada CNN creados al ajustar los parámetros de función de madre Wavelet y valor de la escala de la CWT.

Para una interacción eficiente entre los subsistemas antes mencionados con el usuario y lograr un control en tiempo real con la simulación de una prótesis, se diseñará una Interfaz Gráfica (GUI) de señales EMG. Para ello, primero se desarrollará los principios de diseño de un GUI

propuestos por (Martinez, 2011) orientándolos a los fines de la tesis; más adelante, se programará mediante hilos de ejecución: la adquisición, visualización, procesamiento, entrenamiento y clasificación de las señales EMG.

El último paso del capítulo es la implementación de la simulación de una prótesis de mano como un sistema de retroalimentación visual para poder realizar las pruebas de destreza que validen el sistema de control mioeléctrico propuesto. Debido que el objetivo de la tesis no está enfocado en el diseño de una prótesis, se seleccionará el diseño de una prótesis o mano robótica de libre acceso para poder realizar su simulación en un entorno virtual. Después de la selección, se analizará el diseño de los dedos mediales y del dedo pulgar para comprender sus mecanismos de transmisión de movimiento; asimismo se analizará cinemáticamente el diseño mediante la convección Denavit-Hartenberg con el fin de realizar un correcto ensamblaje de las piezas en el entorno virtual. Finalmente, se implementará la comunicación de la simulación con la GUI mediante *sockets* en una misma red local, para que un usuario con los electrodos EMG conectados y utilizando la GUI, pueda tener una retroalimentación visual al observar el movimiento manual estimado por el controlador en la simulación en tiempo real.

En el siguiente capítulo de la tesis se ejecutará las pruebas de destreza con el propósito de validar el sistema y la metodología desarrollada en este capítulo

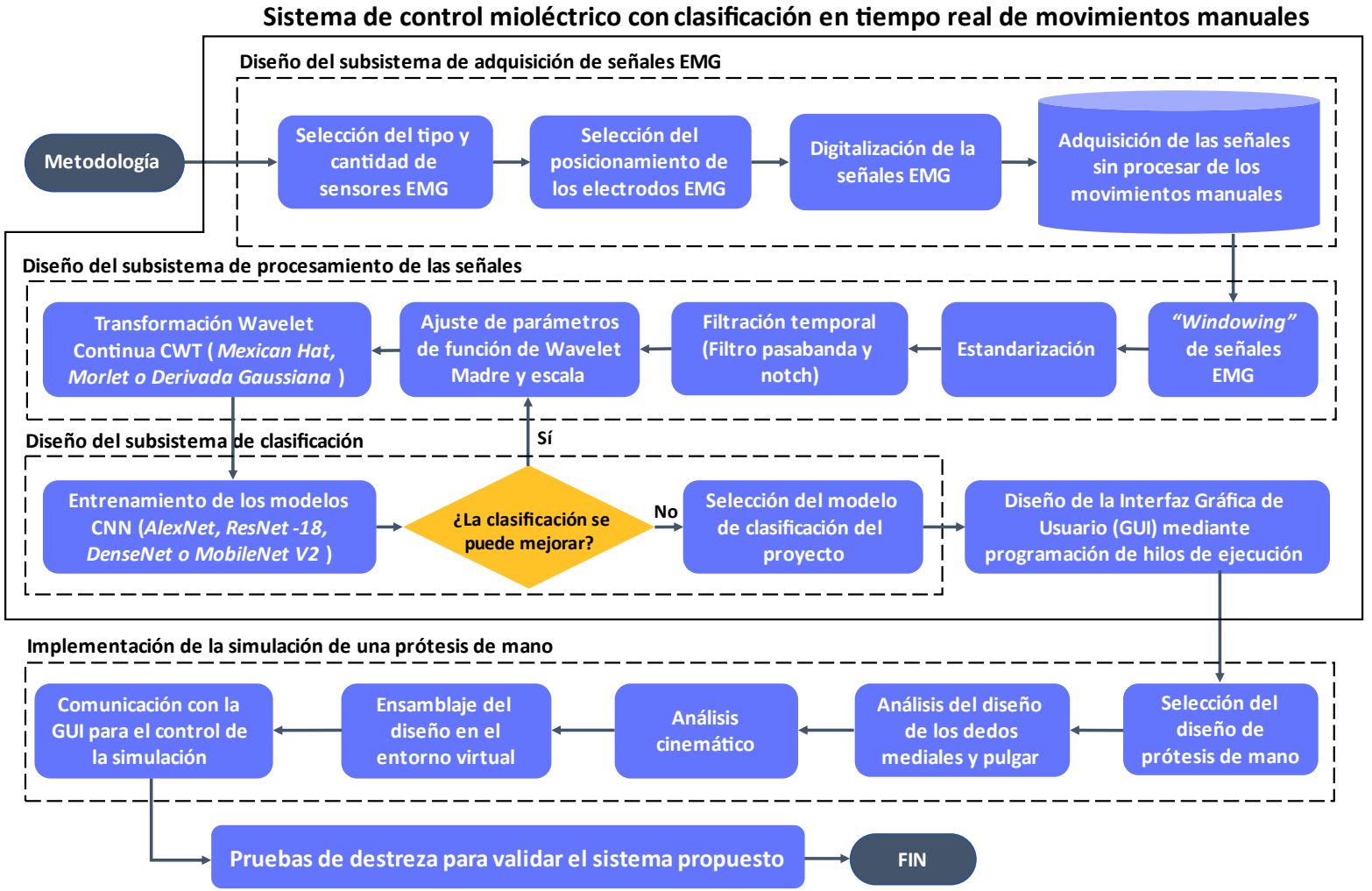


Figura 3.1. Flujograma del desarrollo de la tesis.

Fuente: Elaboración propia

3.2. Diseño del subsistema de adquisición

Para poder adquirir, visualizar y manipular correctamente los datos EMG es importante diseñar correctamente un subsistema de adquisición conformado por elementos y procesos que cumplan los criterios dados por el estado del arte.

En primer lugar, se realizará una correcta selección del sensor EMG en base a criterios técnicos, dimensionales y económicos. Por ejemplo, un criterio importante es el tipo de electrodos debido a su influencia en la comodidad de uso y la portabilidad por un largo plazo, requisitos necesarios para aplicaciones como las prótesis (Fu et al., 2020).

En segundo lugar, se seleccionará la cantidad de sensores EMG la cual influye directamente en la precisión de clasificación y el costo del subsistema de adquisición (Hargrove et al., 2007).

En tercer lugar, se realizará el proceso de posicionamiento de los electrodos en el antebrazo tomando como referencia las recomendaciones del libro de (Leis & Trapani, 2000).

En cuarto lugar se definirá la digitalización de las señales teniendo en cuenta que la cuantificación posea el número bits necesarios para que las señales adquiridas tengan una resolución aceptable y evitar pérdida de información (Zhu et al., 2021).

Por último, se realizará la adquisición de las señales EMG por cada movimiento manual definido por la taxonomía de sujeción de Cutkosky y la clasificación de (Vergara et al., 2012).

3.2.1. Selección del sensor EMG

La señal EMG no amplificada tiene carga típica de 2-3 milivoltios cuando se lee en la piel, por ello se sugiere el uso de amplificadores EMG con un factor de amplificación de al menos 500 para magnificar las señales en el rango de los voltios (Konrad, 2006), es decir que el rango de la salida del sensor debe ser al menos 0-1.5 voltios.

Otro criterio técnico importante para la selección del sensor EMG es la capacidad de frecuencia de muestreo debido a que permite traducir con precisión el espectro de frecuencia de una señal, resultando en un efecto substancial en el rendimiento de la clasificación en tiempo real (Phinyomark et al., 2018).

La potencia de las señales EMG se encuentran entre 10 y 250 Hz, y las recomendaciones científicas (SENIAM, ISEK) sugieren una configuración de banda de amplificación entre 10 y 500 Hz, debido a que la frecuencia de muestreo debería ser por lo menos dos veces la frecuencia de banda de la señal EMG (Konrad, 2006); por tal motivo, el sensor seleccionado debe tener la capacidad de una frecuencia de muestreo igualando o superando los 1000 Hz debido al teorema de muestreo de Nyquist descrito en la ecuación 3.1. Este teorema menciona que la frecuencia de muestreo (F_s) debe ser el doble de la frecuencia mayor ($F_{máx}$) para que la señal analógica sea reconstruida y evitar el efecto *Aliasing* producido en las señales continuas al ser muestreados digitalmente (Konrad, 2006).

$$F_s \geq 2F_{máx} \tag{3.1}$$

Basado en los criterios anteriormente descritos y otros criterios como el tamaño, peso, costo y tipo de electrodos, se realizó una comparación de sensores EMG comerciales en la tabla 3.1.

Tabla 3.1. Comparación de sensores EMG comerciales.

Sensor EMG	Sensor EMG Myoware (SparkFun Electronics, Boulder, , CO, USA)	Sensor Grove EMG Detector (Seeed Studio, Shenzhen, China)	Sensor EMG de electrodos secos (Wuxi Sichiray Co., Ltd., China)
Salida	0 - 3.3V o 5V	0 - 3.3V o 5V	0 – 3.0V
F_s ≥ 1k Hz	Sí	Sí	Sí
Tamaño	25 mm x 42 mm	100 mm x 140 mm	25 mm x 48 mm
Peso	25 g	39 g	27 g
Costo	S/. 280	S/. 136	S/. 160
Tipo de electrodos	Superficiales húmedos	Superficiales húmedos	Superficiales secos

Fuente: Elaboración propia

Los 3 sensores EMG mostrados en la tabla 3.1 cumplen los criterios principales que es la salida amplificada en voltios (factor de amplificación mayor a 500) y la capacidad de una frecuencia de muestreo mayor o igual a 1000 Hz. En la tabla 3.1 también se muestra que respecto al costo, peso y tamaño la diferencia no es muy significativa para el desarrollo de la tesis; por lo tanto, la elección final entre los 3 sensores EMG se basará en el tipo de electrodos.

- **Comparación entre los electrodos húmedos y secos.**

Los electrodos de superficie húmedos son ampliamente utilizados para la adquisición de señales EMG por sus excelentes propiedades eléctricas; no obstante, las molestias, inestabilidad y problemas de infección e irritación de la piel derivados del uso del gel conductor no puede satisfacer la necesidad de adquirir señales EMG por un largo plazo en aplicaciones portátiles como las prótesis.

Adicionalmente el uso de electrodos húmedos requiere un tiempo para la preparación de la piel, como corte de vellos, eliminación de suciedad y recubrimiento de gel electrolítico (Fu et al., 2020).

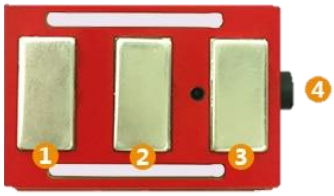
Por otro lado, los electrodos secos, al no utilizar geles conductores, tienen ventajas en comodidad de uso y portabilidad por un largo plazo. A pesar de que las señales adquiridas por los electrodos secos tienen mayor impedancia, aleatoriedad y ruido; la variedad de materiales y el avance de la electrónica integrada ha permitido que los electrodos secos sean ampliamente utilizados en la adquisición de señales EMG (Fu et al., 2020).

Particularmente el posicionamiento de los electrodos secos del sensor EMG de electrodos secos de Wuxi Sichiray Co. no tiene mucha dificultad, teniendo como único requisito que la placa de electrodos se posicione en la misma dirección del músculo.

En síntesis, por criterios de comodidad de uso y portabilidad que se deriva del criterio de tipo de electrodos, se selecciona el sensor EMG de electrodos secos de la empresa Wuxi Sichiray Co.

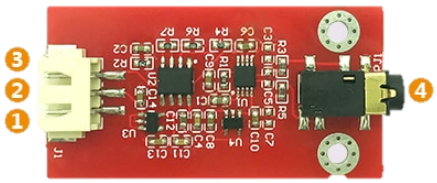
- **Sensor EMG de electrodos secos de Wuxi Sichiray Co.**

El sensor EMG de electrodos secos (Wuxi Sichiray Co., Ltd., China) es un sensor económico conformado por una placa de electrodos y por una placa de filtrado y amplificación como se observa en la figura 3.2. Los datos de fabricante del sensor se pueden ver en el ANEXO C.1.



Número	Definición del Pin
1	Electrodo seco DRY +
2	Electrodo de referencia GND
3	Electrodo seco DRY -
4	Conector PJ-342

(a)



Número	Definición del Pin
1	GND
2	Vcc (3.3 ~ 5.5V)
3	Salida analógica (0 ~ 3.0V)
4	Conector PJ-342

(b)

Figura 3.2. (a) Placa de electrodos (b) Placa de filtrado y amplificación.
Fuente: Wuxi Sichiray Co., LTD

El sensor EMG de electrodos secos (Wuxi Sichiray Co., Ltd., China) adquiere las señales EMG que se encuentran en el rango de los milivoltios y luego estas señales son filtrarlos y amplificados analógicamente hasta el rango de 0 - 3.0V con referencia de voltaje en el 1.5V, tal como se observa en la figura 3.3.

Figura 3.3. Señal en el sensor EMG de Wuxi Sichiray Co.
Fuente: Elaboración propia

3.2.2. Selección de la cantidad de sensores EMG

La cantidad de sensores en el subsistema de adquisición puede intervenir en el rendimiento de la clasificación de movimientos. Por ejemplo, en el trabajo de (Hargrove et al., 2007) se realizó una comparación la precisión de clasificación de 6 patrones en función del número de canales EMG en el antebrazo con dos métodos: El primero es el método simétrico, dónde los canales se posicionaron igualmente

espaciado alrededor de la circunferencia del antebrazo; y el segundo método es el método óptimo, dónde se procesó cada combinación posible de canales dónde se proporcioné mayor precisión.

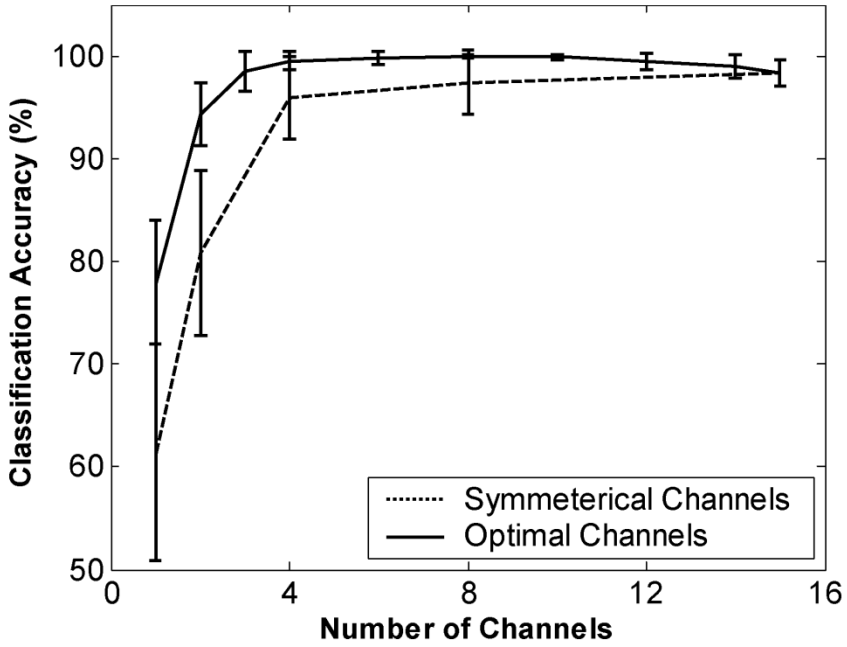


Figura 3.4. Precisión en función al número de canales.
Fuente: Hargrove et al. (2007) Accuracy depending on the number of channels

En la figura 3.4 se muestra que en ambos métodos la precisión de clasificación aumenta con el número de canales EMG, siendo a partir de 3 canales dónde la precisión con el método óptimo es mayor a 95%. Es por ese motivo que se decidió utilizar 3 sensores presumiendo que se obtendrá una alta precisión con un número bajo de sensores, lo cual significaría un menor costo.

3.2.3. Posicionamiento de los electrodos

El posicionamiento de los electrodos también es un proceso importante del subsistema de adquisición porque permite obtener estimaciones precisas y repetibles de las señales EMG en el tiempo (Rainoldi et al., 2004).

Los electrodos secos del sensores EMG de Wuxi Sichiray Co. deben ser posicionados en la misma dirección de los músculos dónde la señal EMG sea distinguible, para ello se utiliza las recomendaciones del libro titulado *"Atlas of Electromyography"* de (Leis & Trapani, 2000) donde se menciona al posicionamiento adecuado de los electrodos para adquirir señales EMG según el músculo.

Cómo se mencionó en el capítulo 2.1.2, en el trabajo de (Alzate Arias, 2018) se destacan los siguientes músculos del antebrazo dedicados al movimiento de los dedos: el músculo extensor común de los dedos, el músculo extensor largo del pulgar, el músculo flexor largo del pulgar, el músculo flexor común superficial de los dedos, el músculo palmar largo y el músculo extensor propio del meñique. Sin embargo, debido a que la cantidad de sensores a utilizar en la investigación son 3, se selecciona los siguientes músculos con sus respectivas posiciones en el antebrazo:

- **Músculos extensores común de los dedos y largo del pulgar**

Se selecciona estos músculos debido a que están relacionados con la extensión de los dedos, además, como se observa en la figura 3.5, las posiciones de los electrodos son cercanas. En consecuencia, una placa de electrodos del sensor EMG podría cubrir ambas áreas. Esta posición es aproximadamente a la mitad del antebrazo (Leis & Trapani, 2000).

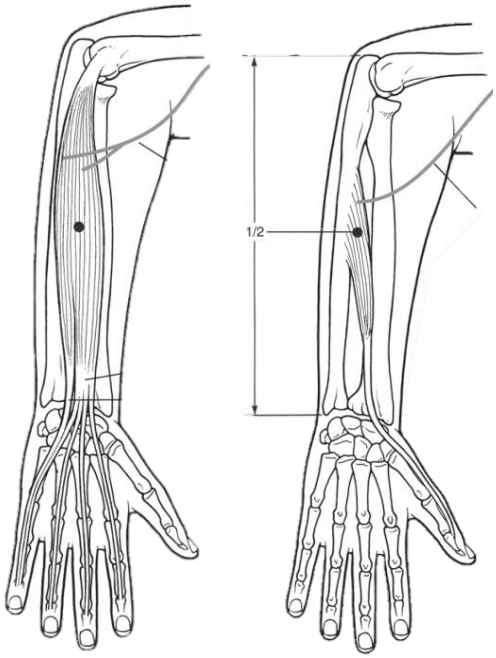


Figura 3.5. (a) Posicionamiento en los músculos extensores común de los dedos y largo del pulgar.
Fuente: Leis & Trapani (2000)

- **Músculo flexor largo del pulgar**

Se selecciona este músculo debido a que permite adquirir señales EMG del movimiento del pulgar siendo parcialmente independiente de los otros dedos. El posicionamiento de los electrodos en el músculo flexor largo del dedo se muestra en la figura 3.6 (Leis & Trapani, 2000).

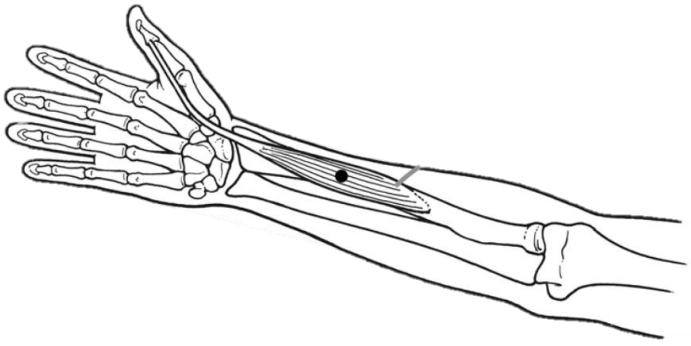


Figura 3.6. Posicionamiento en el músculo flexor largo del pulgar.
Fuente: Leis & Trapani (2000)

- **Músculo flexor superficial de los dedos**

Se selecciona este músculo debido a su relación con la flexión de los dedos mediales. El posicionamiento de los electrodos en el músculo flexor largo del dedo se observa en la Figura 3.7 (Leis & Trapani, 2000).

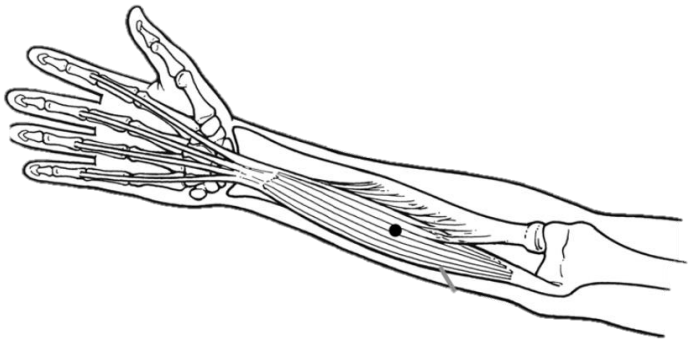


Figura 3.7. Posicionamiento en el músculo flexor superficial de los dedos.
Fuente: Leis & Trapani (2000)

En la figura 3.8 se observa el posicionamiento de las 3 placas de electrodos secos de los sensores EMG Wuxi Sichiray Co. en el antebrazo de una persona para la adquisición de señales.

(a) (b) (c)

Figura 3.8. (a) Músculo extensores largo del pulgar y común de los dedos. (b) Músculo flexor largo del pulgar. (c) Músculo flexor superficial de los dedos.
Fuente: Elaboración propia

3.2.4. Digitalización de las señales EMG

Para poder procesar las señales adquiridas a través de los sensores se realiza la digitalización mediante los conversores analógico-digital (ADC) de la tarjeta Raspberry Pi Pico. Esto debido a que cuenta con tres entradas analógicas de 12 bits como se observa en la figura 3.9 con un voltaje máximo de entrada de 3.3V y con una frecuencia máxima de conversión analógica digital de 500 KHz (Alfareme & Everard, 2021), lo cual es claramente superior al 1 KHz de frecuencia de muestreo para la adquisición de señales. En el ANEXO C.2 se observa los datos de fabricante de la tarjeta.

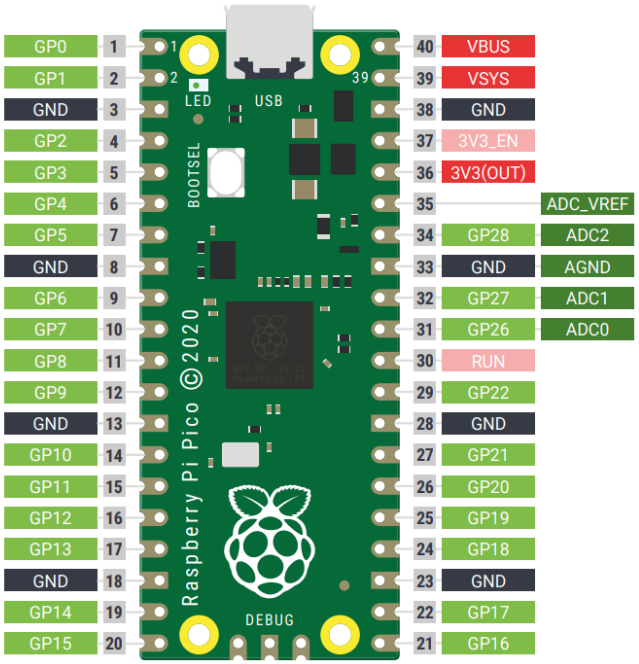


Figura 3.9. Los pines del Raspberry Pi Pico.
Fuente: Halfacree & Everard (2021) The Raspberry Pi Pico' pins

El Raspberry Pi Pico es una tarjeta económica que utiliza el microcontrolador RP2040 con el procesador Arm Cortex M0+. Esta tarjeta permite la programación de software mediante C SDK, C++ o MicroPython (Halfacree & Everard, 2021).

La programación de la placa también se puede realizar en el IDE de Arduino (Earle F., 2022). Para el desarrollo de la digitalización se decide utilizar el IDE de Arduino por su programación de forma simple y rápida en base a sus distintas librerías y el gran soporte de la comunidad (Peña, 2020).

Los tres sensores EMG de electrodos secos de Wuxi Sichiray Co. se conectan en las 3 entradas analógicas para poder digitalizar las señales EMG. Estos datos son enviados mediante comunicación serial hacia una computadora como se observa en la Figura 3.10 mediante un cable Micro USB B con el fin de visualizar y procesar las señales. La velocidad de comunicación serial seleccionada es 115200 baudios con 8 bits para enviar los datos.

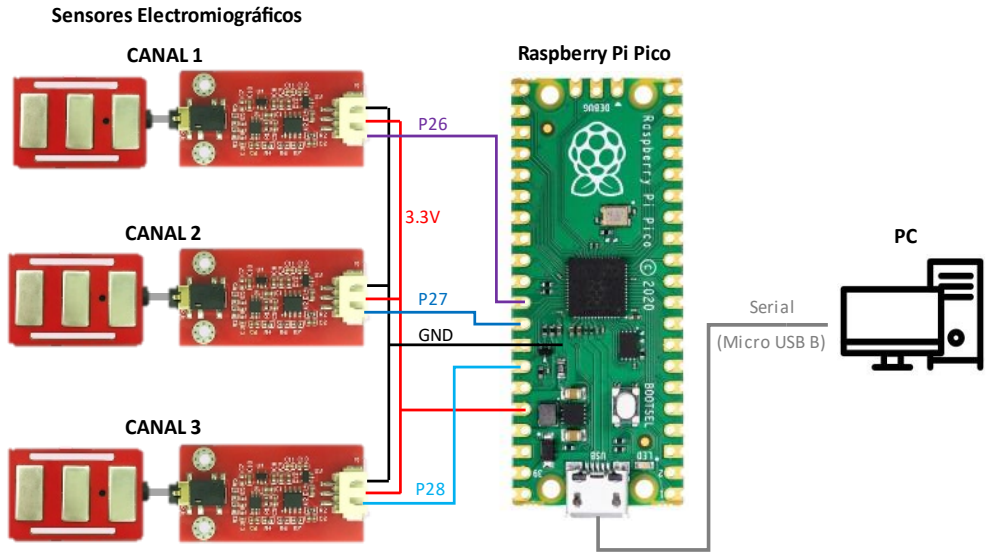


Figura 3.10. Subsistema de adquisición de las señales EMG.
Fuente: Elaboración propia

En la table 3.2 se muestra la relación entre la terminología de los canales EMG y los músculos del antebrazo donde se posicionan los electrodos EMG.

Tabla 3.2. Relación entre los canales EMG y los músculos.

CANAL EMG	Músculo(s) del antebrazo
CANAL 1	Músculo extensor común de los dedos y largo del pulgar.
CANAL 2	Músculo flexor del pulgar
CANAL 3	Músculo superficial de los dedos

Fuente: Elaboración propia

A partir de pruebas de envío por comunicación serial de 3 canales de 12 bits de los ADC de la tarjeta Raspberry Pico con una frecuencia de 1 KHz, se ha notado errores de envío de datos cada cierto tiempo. Por ello, se decidió redimensionamiento los datos a 10 bits antes del envío por comunicación serial. Los 10 bits por canal (resolución de 3.22 mV) continúa siendo aceptable en el estado de arte referente a la adquisición de señales EMG (Zhu et al., 2021).

3.2.5. Adquisición de las señales EMG

En el subcapítulo 2.2.5 se menciona que los movimientos manuales de la taxonomía de Cutkosky aumentan la destreza de las prótesis de ser correctamente clasificadas. De ello, se seleccionaron 4 movimientos manuales que se pertenezcan a la taxonomía de movimientos de agarre (Vergara et al., 2012), los cuales tienen una relación directa con las actividades de vida diaria (AVD). Estos son: empuje de plataforma (movimiento no prensil), pulgar aducido (agarre palmar oblicuo), pinza lateral y pinza pulgar-índice. Adicionalmente se añade la posición de mano relajada como parte de los movimientos a clasificar representando una posición neutral dónde los músculos se encuentran en una posición natural (Alzate Arias, 2018).

En la tabla 3.3 se muestra la taxonomía que se utilizará a lo largo del proyecto para referirse a los movimientos manuales a clasificar.

Tabla 3.3. Taxonomía de los movimientos manuales a clasificar.

Movimiento Manual	Taxonomía
Mano relajada	ReHand
Empuje de plataforma	PlPush
Pulgar aducido	AdThumb
Pinza lateral	LaPinch
Pinza pulgar-índice	ThIndex

Fuente: Elaboración propia

Las señales EMG se adquirieron durante 60 segundos por cada movimiento manual mediante el subsistema de adquisición mostrada en la figura 3.10 y con las 3 placas de electrodos posicionados según la tabla 3.2. Debido a que la frecuencia de muestreo de mil Hertz, se tiene 60000 datos por cada movimiento manual los cuales fueron guardadas en archivos CSV, para su posterior procesamiento y clasificación. El código de la adquisición se muestra en el ANEXO A.1.

A continuación, cada movimiento manual con sus respectivas señales EMG obtenidas mediante el subsistema de adquisición planteado es mostrado:

- **Mano relajada**

El primer movimiento manual a clasificar es la mano relajada, el cual consiste una ligera contracción muscular de los dedos mediales y el pulgar (Calderon et al., 2016). El movimiento manual y las señales EMG que se obtienen mediante el subsistema de adquisición se muestran en la figura 3.11.

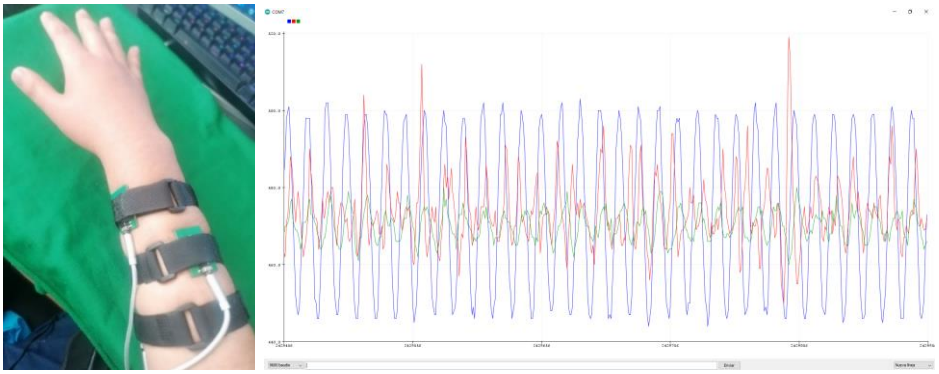


Figura 3.11. Señales EMG del movimiento mano relajada.
Fuente: Elaboración propia

- **Empuje de plataforma**

El segundo movimiento manual a clasificar es el empuje de plataforma, el cual consiste en el movimiento de la extensión de todos los dedos para interactuar con un objeto sin sujetarlo (Vergara et al., 2012). El movimiento manual y las señales EMG que se obtienen mediante el subsistema de adquisición se muestran en la figura 3.12.

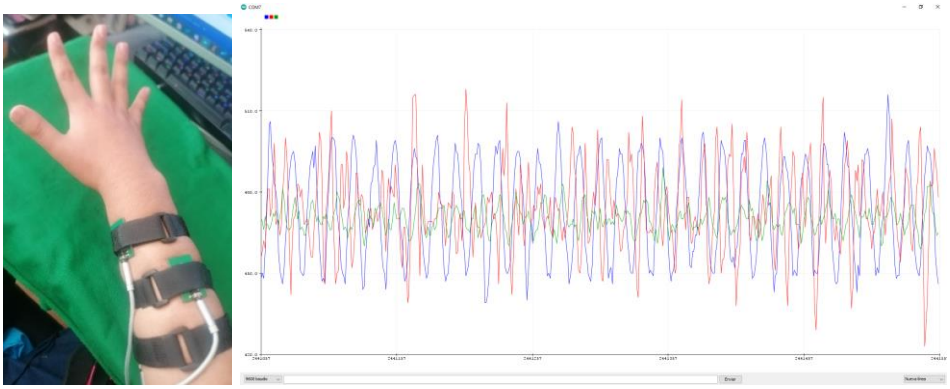


Figura 3.12. Señales EMG del movimiento empuje de plataforma.
Fuente: Elaboración propia

- **Pulgar aducido**

El tercer movimiento manual es el pulgar aducido, el cual está conformado por la extensión abducción del dedo pulgar y la flexión de los dedos mediales. En el agarre interviene los dedos mediales y la palma (Vergara et al., 2012). El movimiento manual y las señales EMG

que se obtienen mediante el subsistema de adquisición se muestran en la figura 3.13.

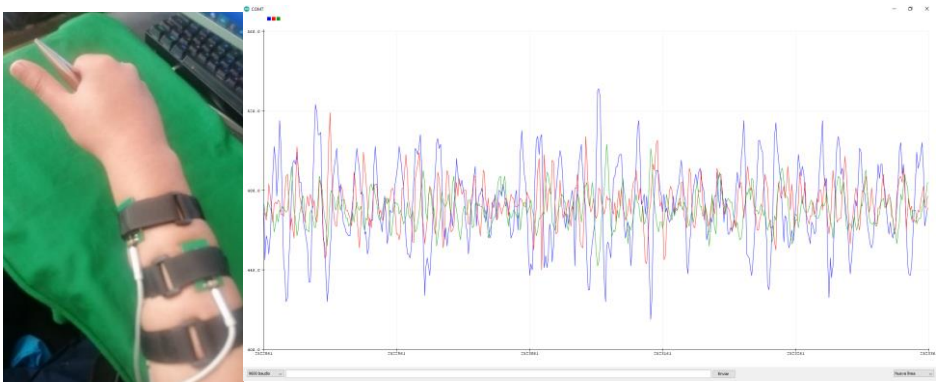


Figura 3.13. Señales EMG del movimiento pulgar aducido.
Fuente: Elaboración propia

- **Pinza lateral**

El cuarto movimiento manual es la pinza lateral, el cual está formado por la flexión de los dedos mediales mientras que el dedo pulgar está en flexión y abducción. El agarre se realiza con la parte lateral del índice y el pulgar (Vergara et al., 2012). El movimiento manual y las señales EMG que se obtienen mediante el subsistema de adquisición se muestran en la figura 3.14.

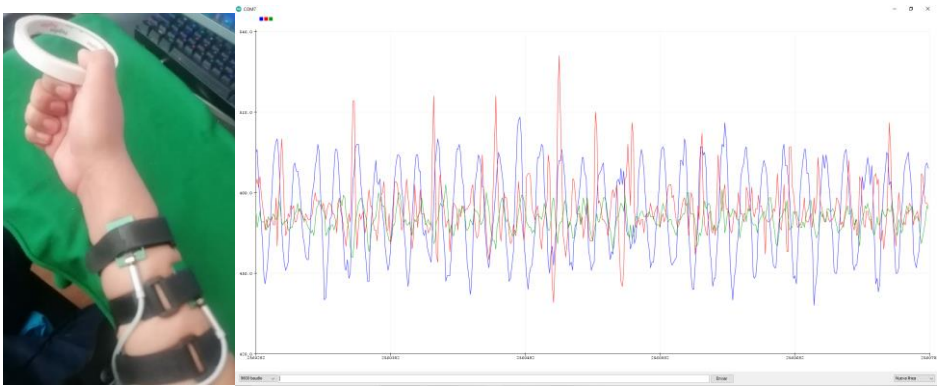


Figura 3.14. Señales EMG del movimiento la pinza lateral.
Fuente: Elaboración propia

- **Pinza pulgar-índice**

El quinto movimiento manual es la pinza pulgar-índice, el cual consiste principalmente en la flexión del dedo pulgar y del dedo índice, donde las falanges distales de estos dedos coinciden en el agarre de objetos pequeños (Vergara et al., 2012). El movimiento manual y las señales EMG que se obtienen mediante el subsistema de adquisición se muestran en la figura 3.15.

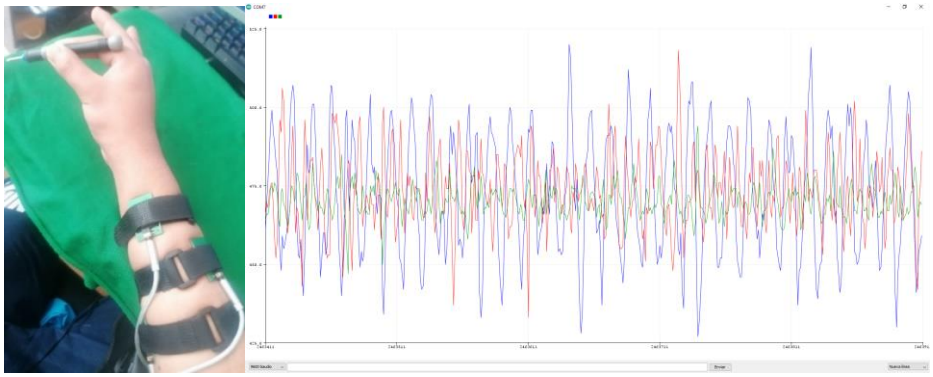


Figura 3.15. Señales EMG del movimiento pinza pulgar-índice.
Fuente: Elaboración propia

3.2.6. Conclusión del subsistema de adquisición

El subsistema de adquisición ha tenido un correcto diseño debido a que los elementos y procesos que lo conforman cumplen con los criterios dados por el estado del arte. Por ejemplo, la selección del sensor EMG y del número de canales han seguido criterios en base a trabajos anteriores y libros dónde se realiza recomendaciones sobre el factor de amplificación, capacidad de frecuencia de muestreo, tipo de electrodos, número de canales, entre otros. El proceso de posicionamiento de los electrodos y la digitalización de las señales también han seguido recomendaciones dadas por libros

especializados. Todo ello es reflejado en las figuras 3.11 hasta 3.15; debido a que las señales EMG adquiridas por cada movimiento manual son distinguibles entre sí. Ello faculta a presumir que, después del procesamiento, se obtendrán escalogramas con un alto potencial de precisión de clasificación.

3.3. Diseño del subsistema de procesamiento

Para poder obtener escalogramas favorables en el entrenamiento de clasificadores de imágenes, es necesario que las señales EMG sean procesadas mediante la técnica *Windowing*, la estandarización, la filtración temporal y la Transformada Wavelet Continua (CWT) (Nahid et al., 2020a). En la figura 3.16 se muestra la secuencia del subsistema de procesamiento propuesto.

En primer lugar, la técnica *Windowing* se utilizará para dividir los 60000 datos por cada movimiento manual obtenidos en el subsistema de adquisición en lapsos llamados ventanas (Preece et al., 2009).

En segundo lugar, la técnica de la estandarización permitirá que los datos de las ventanas tengan el valor promedio centrado en 0 y los valores máximos de cada sensor sean uniformes con el fin de aumentar la precisión de la clasificación (Aceves, 2019).

En tercer lugar, se realizará una filtración temporal; primero con un filtro pasabanda para atenuar los ruidos de compensación de corriente continua, artefactos de movimiento y ruidos de alta frecuencia (Muceli et al., 2010); y seguidamente se utilizará un filtro notch para atenuar el ruido proveniente de la red eléctrica (Kawano & Koganezawa, 2016).

Las señales filtradas serán transformadas en escalogramas mediante la Transformada Wavelet Continua (CWT) variando los parámetros de función de wavelet madre $h(t)$ y el valor de la escala $\boldsymbol{s}$ (Too et al., 2018).

Finalmente se redimensionará los escalogramas a 224 x 244 px para ser entradas eficientes de los modelos de clasificación basados en Deep Learning (Talebi & Milanfar, 2021).

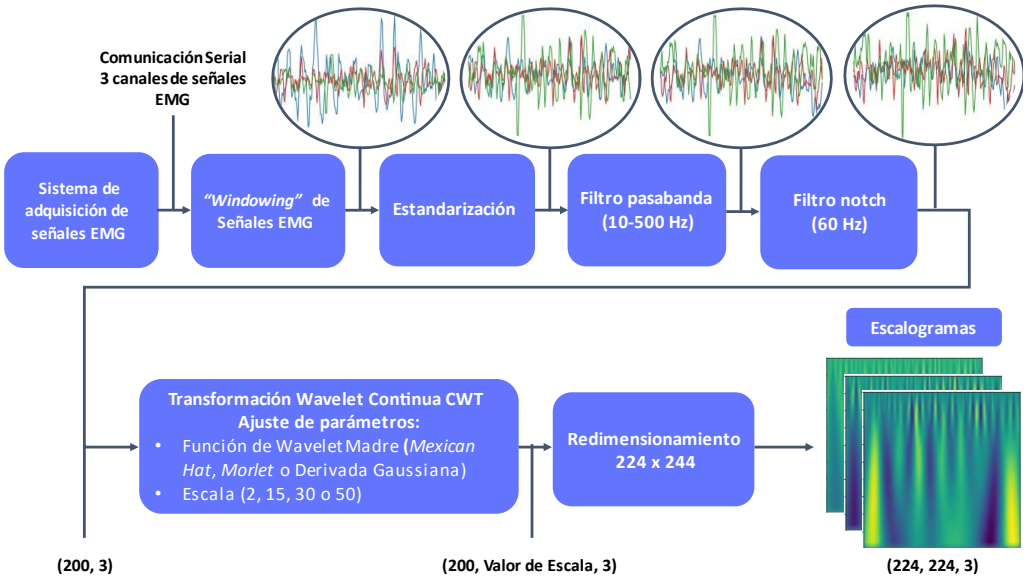


Figura 3.16. Secuencia del subsistema del procesamiento de las señales.
Fuente: Elaboración propia

3.3.1. *Windowing*

La técnica *Windowing* tiene el objetivo de dividir la señal lapsos de tiempo denominados ventanas (Preece et al., 2009). Para el procesamiento, el tamaño de la ventana es esencial en la precisión de la clasificación de patrones de señales EMG. Una ventana corta podría tener una precisión menor; por otro lado, una ventana grande podría tener un rendimiento bajo con potencial de una clasificación errónea (Kerdegari et al., 2013).

En el trabajo de (Nahid et al., 2020a) se recomienda mantener la latencia menor de 300 ms para tener un correcto rendimiento de control en tiempo real para una prótesis. Por ello se escoge ventanas deslizantes de 200 ms (ver figura 3.17) y el 100 ms restante se reserva para el tiempo de procesamiento y el tiempo de clasificación.

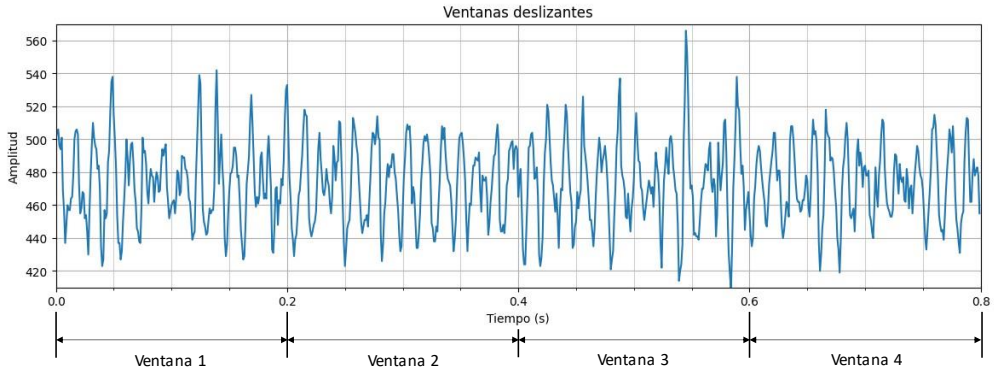


Figura 3.17. Ventanas deslizantes.
Fuente: Elaboración propia

Las primeras ventanas de 200 ms de los 3 canales de las señales al efectuar los 5 movimientos manuales se muestran la figura 3.18. Las columnas y las filas representan los canales y movimientos manuales respectivamente.

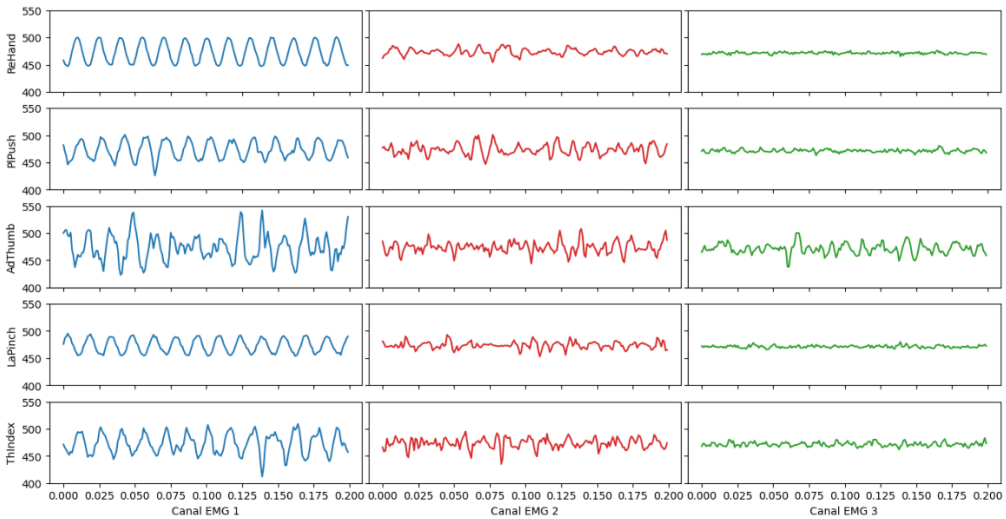


Figura 3.18. Ventanas al realizar los 5 movimientos manuales.
Fuente: Elaboración propia

3.3.2. Estandarización

Parte de las técnicas de procesamiento de las señales fisiológicas es la estandarización de los datos adquiridos, tomando un rol crucial en la mejora de la clasificación de patrones (Aceves, 2019). Esto debido a que el valor promedio, los valores máximos de las señales y la información de las señales en general varían de acuerdo a cada persona y a cada sensor con su respectivo electrodo, por lo que es necesario estandarizar la información a favor de la versatilidad del control de la prótesis (Nazmi et al., 2016).

Las señales obtenidas de los tres sensores EMG Sichiray no tienen el valor promedio en cero como se muestra en la figura 3.18. Asimismo, la variación de los valores máximos según cada sensor EMG es muy notoria. Por lo cual se decide aplicar la estandarización a los tres sensores por separado tomando como datos las señales adquiridas al realizar los 5 movimientos manuales, lo que significa que la media aritmética $\bar{x}$ y la desviación típica σ varían por cada sensor.

Los resultados de las primeras ventanas de los 3 canales de las señales EMG para los 5 movimientos manuales después de la estandarización se observan en la figura 3.19.

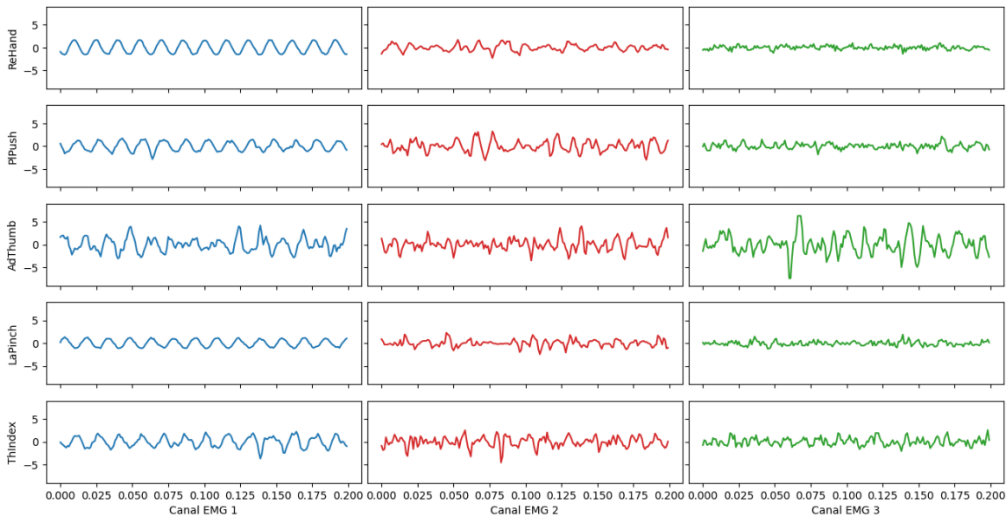


Figura 3.19. Ventanas después de la estandarización.
Fuente: Elaboración propia

3.3.3. Filtro pasabanda

Las recomendaciones del proyecto de la Unión Europea Electromiografía de Superficie para la Evaluación No-Invasiva de los Músculos (SENIAM) y la Sociedad Internacional de Electromiografía y Kinesiología (ISEK) sugieren el uso de filtros pasabanda en el rango de 10-500 Hz para reducir los efectos de *Aliasing* cuando se emplea una frecuencia de muestreo de mil Hertz (Aceves, 2019).

En el trabajo de (Muceli et al., 2010) se utilizó un filtro pasabanda digital Butterworth de cuarto orden para atenuar los ruidos de compensación de corriente continua; los artefactos de movimiento (ruidos que ocurren cuando el sujeto realiza movimientos involuntarios); y los ruidos de alta frecuencia. De igual manera en las investigaciones de (Ting et al., 2019) y (Z. Zhang et al., 2019) se mencionan la predilección del filtro pasabanda Butterworth de cuarto orden para la atenuación de ruidos en las señales EMG.

Por lo tanto, se decide aplicar un filtro pasabanda digital Butterworth de cuarto orden con rango de frecuencias de 10-500 Hz a las señales adquiridas al realizar los 5 movimientos manuales.

Los resultados de las primeras ventanas de los 3 canales de las señales EMG para los 5 movimientos manuales después de la filtración pasabanda se observan en la figura 3.20.

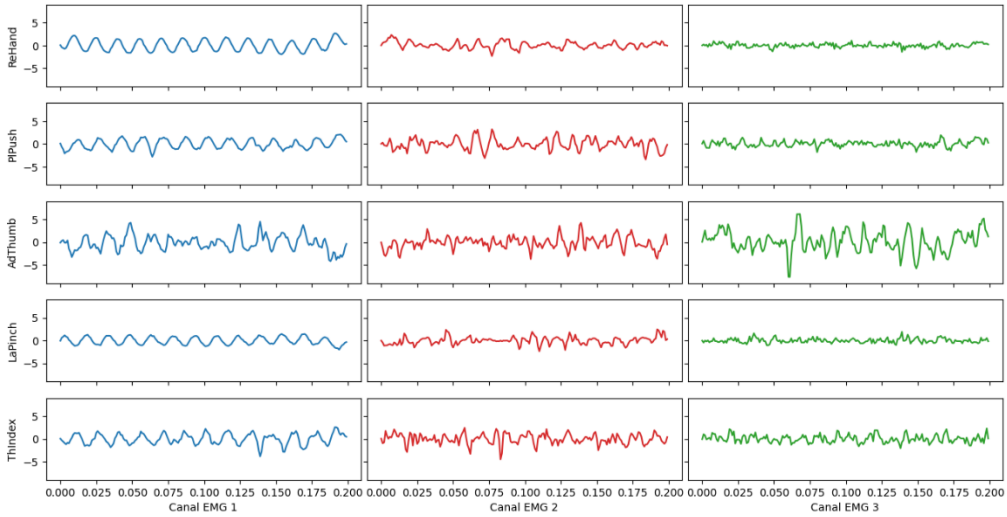


Figura 3.20. Ventanas después del filtro pasabanda.
Fuente: Elaboración propia

3.3.4. Filtro notch

El filtro notch es un tipo de filtro rechaza banda que se utiliza para atenuar el ruido proveniente de la red eléctrica comercial (Kawano & Koganezawa, 2016). La frecuencia seleccionada corresponde a la red eléctrica de cada país. Por ejemplo, en las investigaciones de (Kurapa et al., 2020) y (Turnip et al., 2021) se seleccionaron filtros notch de segundo grado y frecuencia de corte de 50 Hertz para atenuar los ruidos provenientes de la red eléctrica que interfieren en la adquisición de las señales biológicas.

En el Perú, país dónde se realiza la presente investigación, la red eléctrica posee una frecuencia de 60 Hertz (*World plugs*, s. f.), es por ello que se selecciona un filtro notch de segundo grado y frecuencia de corte de 60 Hz.

Los resultados de las primeras ventanas de los 3 canales de las señales electromiográficas para los 5 movimientos manuales después de la filtración notch se observan en la figura 3.21.

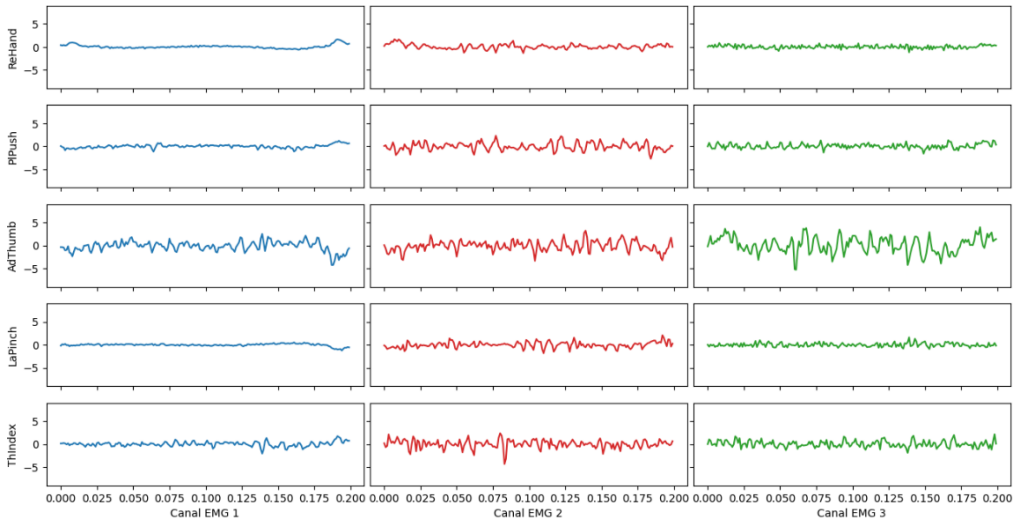


Figura 3.21. Ventanas después del filtro notch.
Fuente: Elaboración propia

3.3.5. Transformada Wavelet Continua

La transformada Wavelet Continua (CWT) permite representar conjuntamente el tiempo y frecuencia a partir de una señal en una imagen denominada escalograma. La CWT se genera a partir de tres parámetros: La función de wavelet madre $h(t)$, el factor de escalamiento $\boldsymbol{s}$ y el factor de desplazamiento $\boldsymbol{\tau}$ (Dorf et al., 2018).

En los últimos años, investigaciones de vanguardia han utilizado la CWT en el procesamiento y clasificación de señales EMG.

Por ejemplo, en el trabajo de (Too et al., 2018) se evaluó 12 diferentes funciones de wavelet madre $h(t)$ de CWT con valores de escala $\boldsymbol{s}$ de 8, 16 y 32 para poder hallar la función wavelet madre con determinada escala que tenga mejor rendimiento en la clasificación de 10 movimientos manuales mediante la adquisición de dos dispositivos EMG portátiles Shimmer (cuatro canales de señales EMG en total) con los electrodos ubicados en el antebrazo.

Otro trabajo similar es de (Kakoty et al., 2015), dónde se evaluó 7 diferentes funciones de wavelet madre $h(t)$ de CWT con valores de índice de escala $\boldsymbol{s}$ de 11, 21, 35, 101 en la clasificación de 6 tipos de agarre usados en actividades de la vida diaria. Los resultados de la investigación se observan en la tabla 3.4, dónde la precisión de la clasificación de los 6 tipos agarre variando los parámetros función de wavelet madre $h(t)$ de CWT y el índice de escala $\boldsymbol{s}$.

En ambas investigaciones el parámetro factor de desplazamiento $\boldsymbol{\tau}$ es igual a 0 debido a que no aporta una mayor precisión a la clasificación de patrones.

Tabla 3.4. Tasas de reconocimiento promedio de 6 tipos de agarre a través de funciones de CWT en diferentes índices de escala.

Wavelet functions	Recognition rate			
	Scale index 11 (%)	Scale index 21 (%)	Scale index 35 (%)	Scale index 101 (%)
Gaussian	79 ± 2.2	80 ± 1.9	80 ± 2.8	81 ± 2.6
Morlet	77 ± 2.8	77 ± 2.7	76 ± 3.0	76 ± 2.8
Meyer	75 ± 2.2	74 ± 2.0	74 ± 2.8	74 ± 2.6
Symlet 4	74 ± 2.4	74 ± 2.2	73 ± 2.6	74 ± 2.2
Db	72 ± 3.2	73 ± 3.0	73 ± 3.0	74 ± 3.4
Mexicanhat	70 ± 2.4	71 ± 2.2	71 ± 2.8	72 ± 2.6
Haar	62 ± 2.2	62 ± 2.1	62 ± 2.4	62 ± 2.2

Fuente: Kakoty et al. (2015) Average recognition rates over six grasp types through CWT functions at different scale index

En el presente trabajo se realiza la misma metodología de las dos investigaciones previamente mencionadas. Las ventanas de las señales EMG después de la estandarización y los filtros temporales son transformadas a escalogramas mediante la CWT seleccionando 5 funciones de wavelet madre $h(t)$ y 4 valores de escala $\boldsymbol{s}$ diferentes**.**

Para realizar la CWT se utiliza el software *"PyWavelets"* para Python debido a que es una librería de Transformación Wavelet de código abierto de alto rendimiento y fácil uso (*PyWavelets Documentation*, s. f.). Por ello, las funciones wavelet madres $h(t)$ seleccionadas están disponibles en *"PyWavelets"* y son las siguientes: *Mexican Hat*, *Morlet*, Primera Derivada Gaussiana, Segunda Derivada Gaussiana y la Tercera Derivada Gaussiana. Por otro lado, los valores de escala seleccionados son $\boldsymbol{s}$: 2,15, 30 y 50.

- **Función wavelet madre *Mexican Hat***

La función wavelet madre *Mexican Hat* se muestra en la ecuación 3.2, dónde el coeficiente es un factor de normalización para que la wavelet tenga energía unitaria (*PyWavelets Documentation*, s. f.). Los escalogramas de las primeras ventanas de los 3 canales de señales EMG para los 5 movimientos manuales cuando la función wavelet madre $h(t)$ es *Mexican Hat* y la escala $\boldsymbol{s}$ es 15 se muestran en la figura 3.22.

$$h(t) = \frac{2}{\sqrt{3}\sqrt[4]{\pi}} exp^{-\frac{t^2}{2}}(1 - t^2) \qquad (3.2)$$

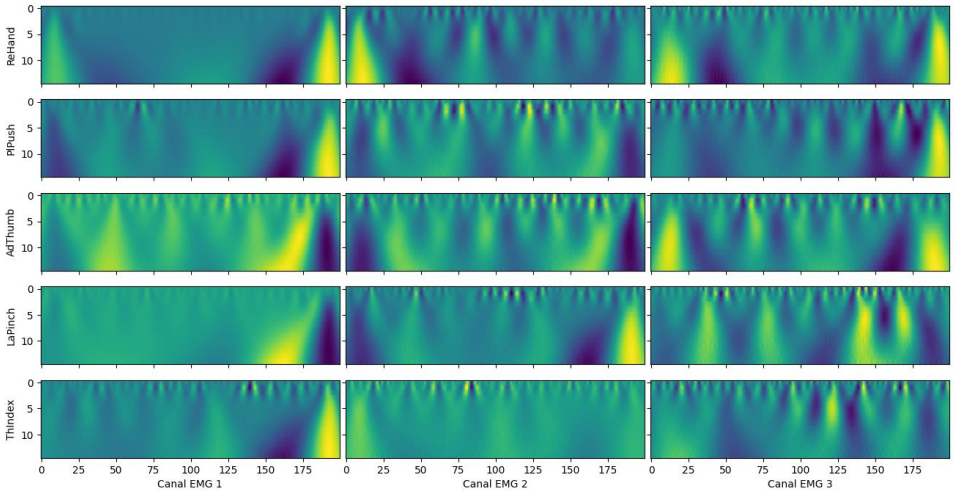


Figura 3.22. Escalogramas con la wavelet madre *Mexican Hat*.
Fuente: Elaboración propia

- **Función wavelet madre *Morlet***

La función wavelet madre *Morlet* se muestra en la ecuación 3.3 (*PyWavelets Documentation*, s. f.). Los escalogramas de las primeras ventanas de los 3 canales de señales EMG para los 5 movimientos manuales cuando la función wavelet madre $h(t)$ es *Morlet* y la escala $\boldsymbol{s}$ es 15 se muestran en la figura 3.23

$$h(t) = exp^{-\frac{t^2}{2}} cos(5t) \qquad (3.3)$$

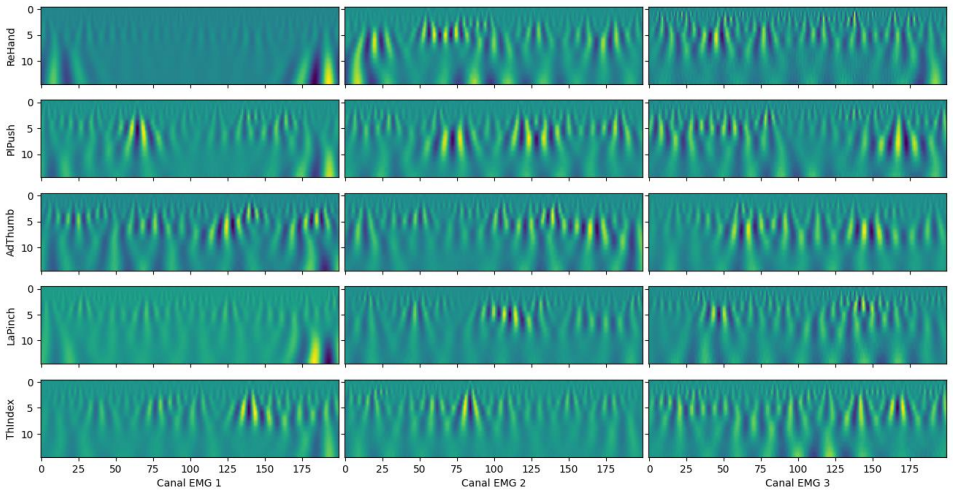


Figura 3.23. Escalogramas con la wavelet madre *Morlet*.
Fuente: Elaboración propia

- **Función wavelet madre Primera Derivada Gaussiana**

La función wavelet madre Primera Derivada Gaussiana se muestra en la ecuación 3.4, dónde el coeficiente C es el factor de normalización para que la función wavelet tenga energía unitaria (*PyWavelets Documentation*, s. f.). Los escalogramas de las primeras ventanas de los 3 canales de señales EMG para los 5 movimientos manuales cuando la función wavelet madre $h(t)$ es la Primera Derivada Gaussiana y la escala $\boldsymbol{s}$ es 15 se muestran en la figura 3.24.

$$h(t) = C\frac{d\left(exp^{-t^2}\right)}{dt} \tag{3.4}$$

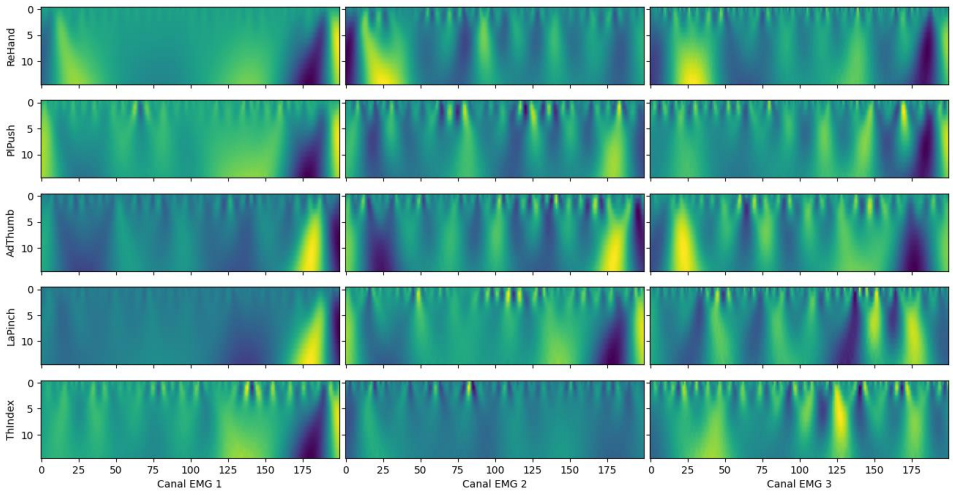


Figura 3.24. Escalogramas con la wavelet madre 1° Derivada Gaussiana.
Fuente: Elaboración propia

- **Función wavelet madre Segunda Derivada Gaussiana**

La función wavelet madre Segunda Derivada Gaussiana se muestra en la ecuación 3.5, dónde el coeficiente C es el factor de normalización para que la función wavelet tenga energía unitaria (*PyWavelets Documentation*, s. f.). Los escalogramas de las primeras ventanas de los 3 canales de señales EMG para los 5 movimientos

manuales cuando la función wavelet madre $h(t)$ es la Segunda Derivada Gaussiana y la escala $\boldsymbol{s}$ es 15 se muestran en la figura 3.25.

$$h(t) = C\frac{d^2\left(exp^{-t^2}\right)}{dt^2} \qquad (3.5)$$

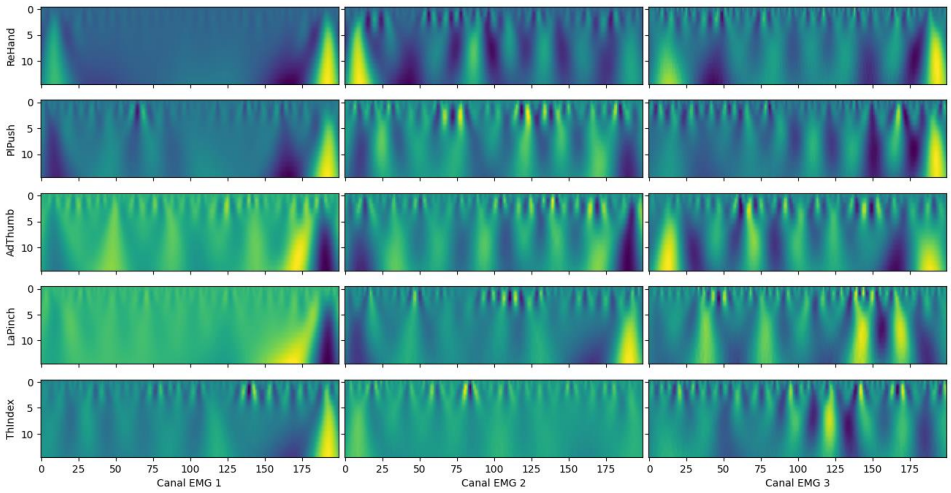


Figura 3.25. Escalogramas con la wavelet madre 2° Derivada Gaussiana.
Fuente: Elaboración propia

- **Función wavelet madre Tercera Derivada Gaussiana**

La función wavelet madre Tercera Derivada Gaussiana se muestra en la ecuación 3.6, dónde el coeficiente C es el factor de normalización para que la función wavelet tenga energía unitaria (*PyWavelets Documentation*, s. f.). Los escalogramas de las primeras ventanas de los 3 canales de señales EMG para los 5 movimientos manuales cuando la función wavelet madre $h(t)$ es la Tercera Derivada Gaussiana y la escala $\boldsymbol{s}$ es 15 se muestran en la figura 3.25.

$$h(t) = C\frac{d^3\left(exp^{-t^2}\right)}{dt^3} \qquad (3.6)$$

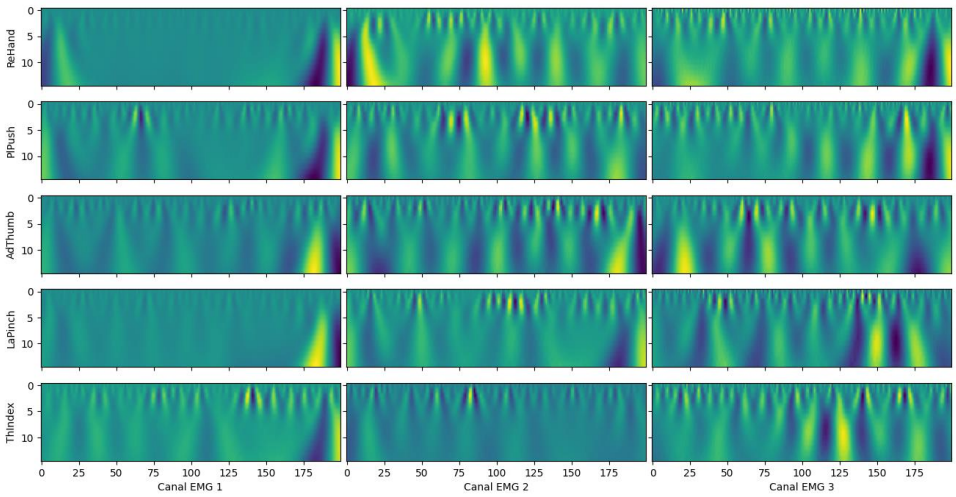


Figura 3.26. Escalogramas con la wavelet madre 3° Derivada Gaussiana.
Fuente: Elaboración propia

Finalmente los escalogramas son redimensionados a una resolución espacial relativamente pequeña como es el clásico 224 x 224 px con el objetivo de ser eficientes en el entrenamiento de los modelos basados en Deep Learning (Talebi & Milanfar, 2021).

3.3.6. Conclusión del subsistema de procesamiento

El subsistema de procesamiento de las señales ha logrado transformar correctamente las señales EMG recibidas en el subsistema de adquisición en escalogramas favorables en el entrenamiento de clasificadores de imágenes. Las técnicas de *Windowing* y estandarización ha permitido dividir las señales en ventanas de 200 ms para luego ser centrada y uniformizadas. El filtro pasabanda y el filtro notch ha permitido atenuar la mayoría de ruidos que se produjeron durante la adquisición. Finalmente, se ha obtenido diversos escalogramas variando los parámetros de la CWT para que, durante el diseño del subsistema de clasificación, se seleccione aquellos que permitan un alto rendimiento de clasificación.

3.4. Diseño del subsistema de clasificación

El subsistema de clasificación tiene como objetivo evaluar y comparar el entrenamiento de 4 arquitecturas de Deep Learning denominadas Redes Neuronales Convolucionales para clasificar los 5 movimientos manuales a partir de las señales EMG obtenidos en el subsistema de adquisición y transformadas en escalogramas en el subsistema de procesamiento tal como se muestra en la figura 3.27.

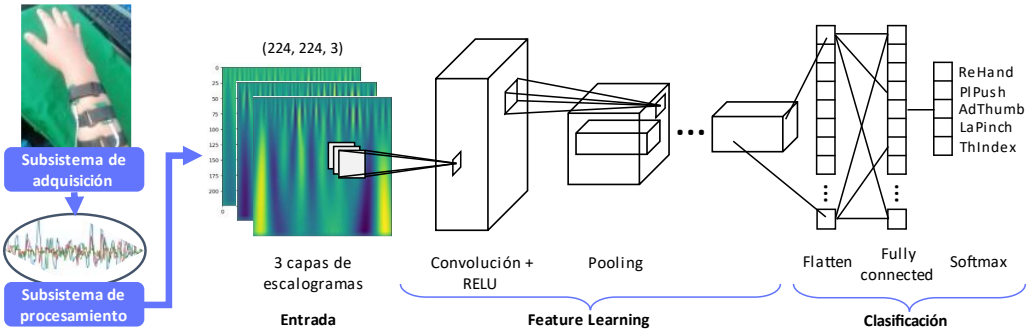


Figura 3.27. Arquitectura general basado en Deep Learning para la clasificación de los 5 movimientos manuales.
Fuente: Elaboración propia

El estado de arte sugiere evaluar diferentes funciones de wavelet madre $h(t)$ y valores de escala $\boldsymbol{s}$ de la CWT para determinar los parámetros que generan el mejor rendimiento en la clasificación de señales EMG (Too et al., 2018) (Kakoty et al., 2015). Es por es tal motivo que cada arquitectura entrenará 20 modelos de clasificación al ajustar los parámetros de la CWT (figura 3.28).

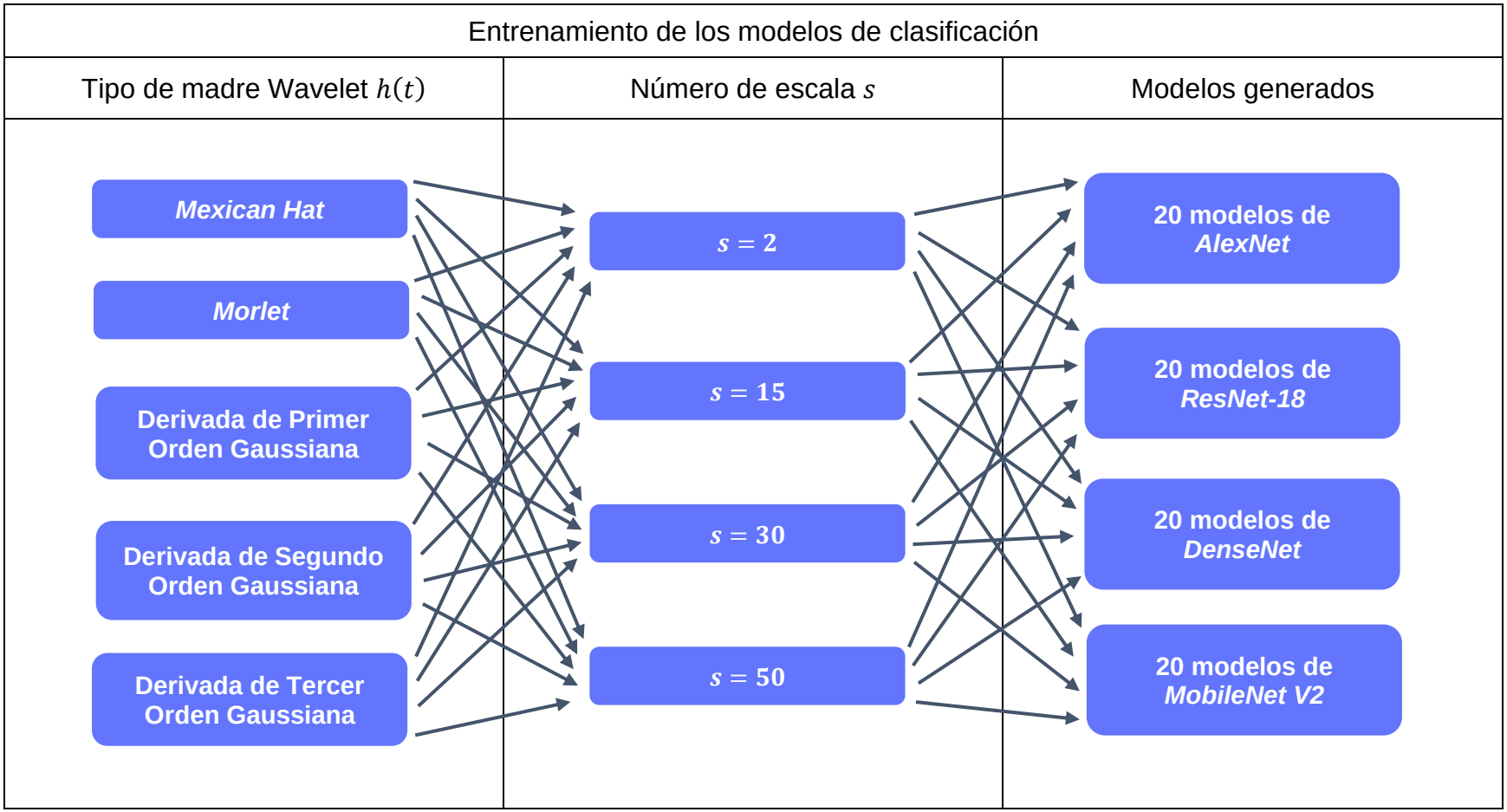


Figura 3.28. Tabla de entrenamiento de los modelos de clasificación.
Fuente: Elaboración propia

Las Redes Neuronales Convolucionales (CNN) seleccionadas son: *AlexNet* (Krizhevsky et al., 2012), *ResNet-18* (He et al., 2015), *DenseNet* (Huang et al., 2018), y MobileNet V2 (Sandler et al., 2019); debido a sus altos rendimientos en la clasificación de imágenes. La descripción y características de cada CNN, se presentarán en el presente subcapítulo.

Durante el entrenamiento de los modelos se aplicará *Transfer Learning*, debido a que permite iniciar el entrenamiento con los pesos de un modelo previamente entrenado con un grupo de millones de imágenes y miles de clases como *ImageNet* (J. Deng et al., 2009), con el objetivo de aumentar el rendimiento de la clasificación reduciendo significativamente el tiempo de entrenamiento (Zhuang et al., 2021). En el presente proyecto, se utilizará las CNN preentrenadas con *ImageNet* reemplazando la última capa por un extractor de características fijas de 5 clases para los 5 movimientos manuales.

Los escalogramas obtenidos previamente se dividen en un 70 % para el entrenamiento y un 30 % para la validación, empleando validación cruzada para asegurar la independencia entre estos dos conjuntos. Las características principales del entrenamiento se observan en la tabla 3.5.

Tabla 3.5. Tabla de características del entrenamiento.

Característica del entrenamiento	Valor
Aprendizaje por transferencia (*transfer learning*)	Sí
Tamaño del lote (*batch size*)	30
Taza de aprendizaje (*learning rate*)	0.0003
Decaimiento de peso (*weight decay*)	0.0001
Optimizador (*optimiser*)	Adam
Función de perdida (*lost function*)	Entropía cruzada (CEL)
Etapas (*epoch*)	3

Fuente: Elaboración propia

Posteriormente al entrenamiento de los modelos, se realizará una comparación de las 80 modelos de clasificación para seleccionar el modelo que tenga el mejor rendimiento, tomando como métricas: la precisión, la pérdida de entropía cruzada (CEL) y el tiempo total (tiempo de procesamiento más tiempo de clasificación).

3.4.1. AlexNet

Unos de los modelos CNN más conocido es la arquitectura *AlexNet* (Krizhevsky et al., 2012) debido a que significó un avance crucial en el Deep Learning junto al campo de la visión por computadora para tareas de clasificación y reconocimiento visual.

En las dos primeras capas de *AlexNet* se realiza Convolución (Conv.) y *Max-pooling* (MXP) con Normalización de Respuesta Local (LRN); luego se tiene tres capas convolucionales con función de activación ReLU; después, 2 capas totalmente conectadas (FC) y finalmente, la capa Soft-max (Alom et al., 2018). La arquitectura AlexNet se muestra en la figura 3.29.

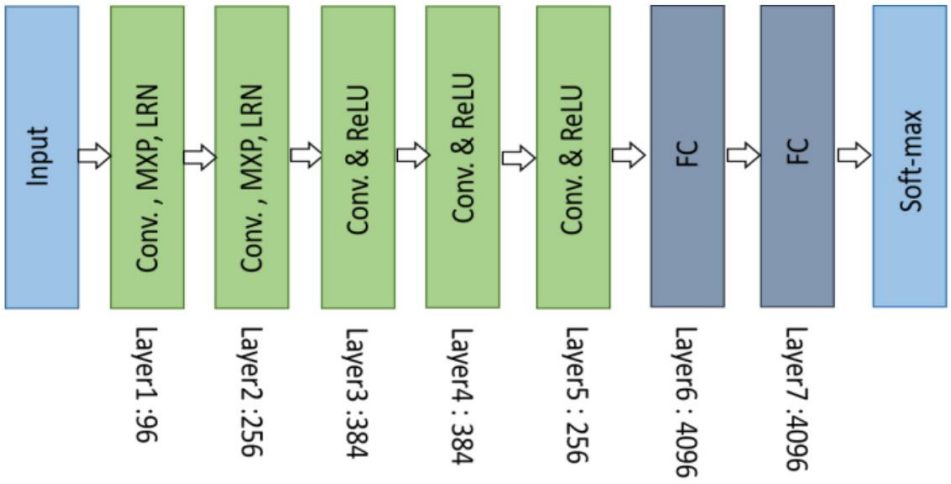


Figura 3.29. Arquitectura de *AlexNet*.
Fuente: Alom et al. (2018) Architecture of AlexNet

En el entrenamiento de la arquitectura *AlexNet* se realiza un proceso llamado paralelización, el cual consiste en repartir el entrenamiento de las neuronas entre dos GPU, con el objetivo de obtener y enviar datos a la memoria de la otra directamente, sin pasar por la memoria de la máquina *host*. Sin embargo, se tiene la particularidad de que las GPU tienen delimitadas las responsabilidades comunicándose solo en ciertas capas. Una GPU ejecuta las capas de la parte superior, mientras que la otra GPU ejecuta las capas de la parte inferior, tal como se muestra en la figura 3.30 (Krizhevsky et al., 2012).

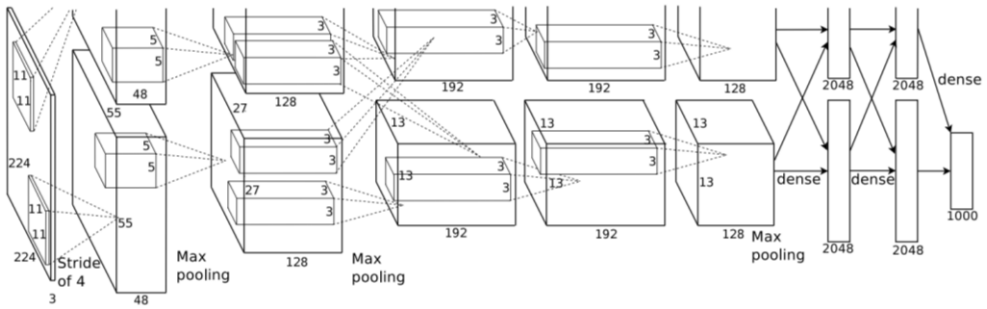


Figura 3.30. Delimitación de responsabilidades entre las dos GPU.
Fuente: Krizhevsky et al. (2012) Delineation of responsibilities between the two GPUs

Los modelos de clasificación de los movimientos manuales generados al variar los parámetros de la CWT (figura 3.28) se entrenan con las características de la tabla 3.5 mediante la CNN *AlexNet*.

La precisión, la pérdida de entropía cruzada (CEL), el tiempo de procesamiento, el tiempo de clasificación y el tiempo total de cada modelo de clasificación después del entrenamiento se muestran la tabla 3.6.

Tabla 3.6. Resultados del entrenamiento con la CNN *AlexNet*.

Función de Wavelet Madre		Número de escala	Precisión	CEL	Tiempo Proces. (ms)	Tiempo Clasif. (ms)	Tiempo total (ms)
Mexican Hat		2	0.8978	0.1827	13.07	12.97	26.04
		15	0.9622	0.1462	15.04	13.01	28.05
		30	0.9311	0.4395	17.97	13.00	30.97
		50	0.9578	0.2747	26.51	12.99	39.50
Morlet		2	0.7244	0.6943	13.97	13.00	26.97
		15	0.9600	0.1342	15.67	12.99	28.66
		30	0.9622	0.3160	19.52	12.99	32.51
		50	0.9533	0.3716	22.98	13.00	35.98
Derivada Gaussiana	Primer Orden	2	0.9156	0.3075	13.38	12.99	26.37
		15	0.9600	0.2133	14.99	12.99	27.98
		30	0.9422	0.1826	17.00	13.00	30.00
		50	0.9778	0.0857	23.01	12.97	35.98
	Segundo Orden	2	0.8378	0.3049	13.24	12.51	25.75
		15	0.9578	0.1664	15.99	13.00	28.99
		30	0.9200	0.2819	17.75	12.99	30.74
		50	0.9400	0.3876	22.99	13.00	35.99
	Tercer Orden	2	0.8689	0.3990	12.99	12.99	25.98
		15	0.9244	0.2288	16.02	13.01	29.03
		30	0.9422	0.2012	18.72	12.99	31.71
		50	0.9622	0.2297	21.99	16.64	35.63

Fuente: Elaboración propia

3.4.2. ResNet-18

ResNet-18 (He et al., 2015) es una arquitectura de 18 capas que nace en la búsqueda de tratar los problemas de la desaparición del gradiente y del sobreajuste en redes de alta profundidad, es decir que cuentan con un gran número de capas. Este trabajo toma gran relevancia debido a que una mayor profundidad de una red, significa una mayor complejidad de las funciones que puede calcular, por lo tanto, una mayor eficiencia de la red (Raghu et al., 2017).

Los autores de *ResNet-18* introdujeron el concepto de *Residual learning,* el cual utiliza bloques residuales de construcción, tal como se observa en la figura 3.31. Un bloque residual está formado por un lado residual (lado izquierdo de la figura 3.31) y una conexión atajo o función identidad (lado derecho de la figura 3.31). De esta forma la información se conserva al omitir su paso por algunas capas permitiendo que la gradiente de la función de pérdida regrese al inicio directamente. Finalmente, para obtener un bloque residual se realiza una sumatoria del lado residual más la función identidad y se aplica la función de activación ReLU (He et al., 2015).

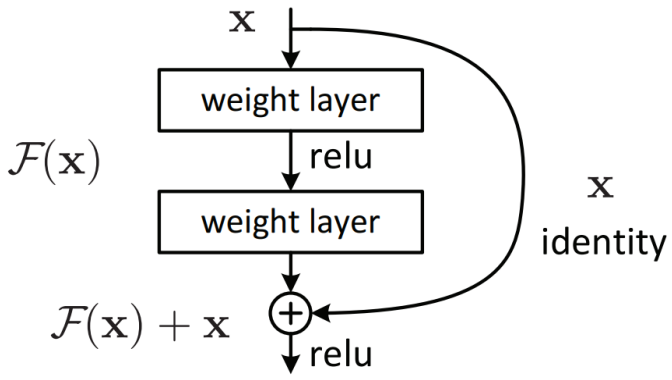


Figura 3.31. Aprendizaje residual: un bloque de construcción.
Fuente: He et al. (2015) Residual Learning: a build block

Los modelos de clasificación de los movimientos manuales generados al variar los parámetros de la CWT (figura 3.28) se entrenan con las características de la tabla 3.5 mediante la CNN *ResNet-18*.

En la tabla 3.7 se muestra la precisión, la pérdida de entropía cruzada (CEL), el tiempo de procesamiento, el tiempo de clasificación y el tiempo total de cada modelo de clasificación después del entrenamiento.

Tabla 3.7. Resultados del entrenamiento con la CNN *ResNet-18*.

Función de Wavelet Madre		Número de escala	Precisión	CEL	Tiempo Proces. (ms)	Tiempo Clasif. (ms)	Tiempo total (ms)
Mexican Hat		2	0.9267	0.1237	13.99	32.03	46.02
		15	0.9578	0.4371	14.99	31.25	46.24
		30	0.9689	0.2280	18.00	30.01	48.01
		50	0.9222	0.3149	23.00	31.99	54.99
Morlet		2	0.6511	1.1932	13.00	32.59	45.59
		15	0.9000	0.7209	15.97	32.30	48.27
		30	0.9578	0.2568	18.00	31.02	49.03
		50	0.9511	0.1131	23.00	30.99	53.99
Derivada Gaussiana	Primer Orden	2	0.9733	0.1229	12.5	32.03	44.53
		15	0.9533	0.1426	14.99	31.99	46.98
		30	0.9667	0.1096	16.99	31.99	48.98
		50	0.9267	0.2984	21.99	30.99	52.98
	Segundo Orden	2	0.9467	0.1320	12.99	32.10	45.09
		15	0.9356	0.2215	14.99	32.02	47.01
		30	0.8444	0.7033	17.00	30.80	47.80
		50	0.9556	0.2266	20.99	29.99	50.99
	Tercer Orden	2	0.7733	0.9973	12.99	31.99	44.98
		15	0.9800	0.1719	15.12	31.53	46.65
		30	0.9667	0.1638	19.00	32.00	51.00
		50	0.9111	0.2173	21.09	31.46	52.55

Fuente: Elaboración propia

3.4.3. DenseNet

La red convolucional *DenseNet* es un tipo de Red Neuronal Convolucional (CNN) caracterizado por conectar cada capa con todas las demás capas de forma retroalimentada para garantizar un altísimo flujo de datos entre las capas. En la figura 3.32 se puede observar la retroalimentación, cada capa recibe información de las capas anteriores y transmite su información a las capas posteriores (Huang et al., 2018).

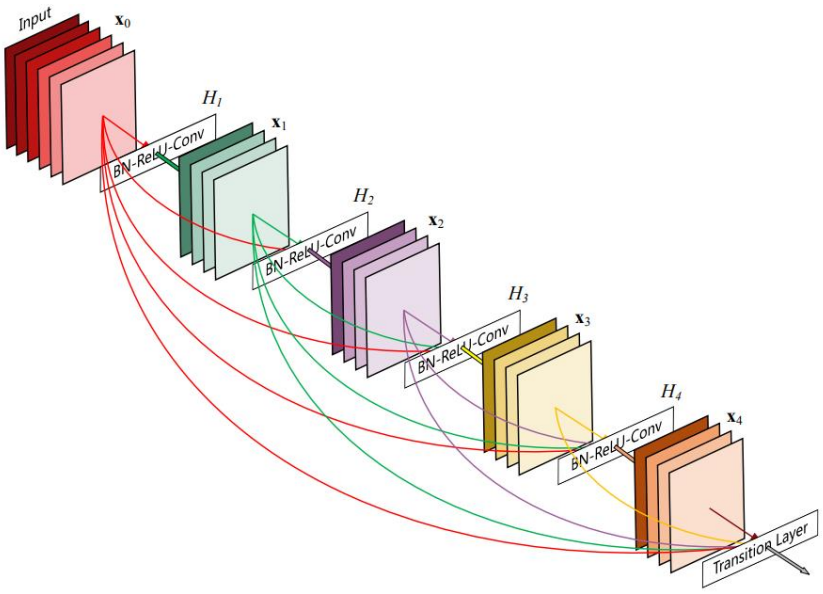


Figura 3.32. Bloque denso de 5 capas donde cada capa toma los mapas característicos anteriores como entrada.

Fuente: Huang et al. (2018) A 5-layer dense block where each layer takes all preceding feature-maps as input

A diferencia de las CNN tradicionales que normalmente tienen *N* capas con *N* conexiones, el *DenseNet* tiene *N(N+1)/2* conexiones directas debido a que la *N*-ésima capa tiene *N* entradas que consiste en los mapas de características anteriores. Este enfoque permite tener varias ventajas como: el fortalecimiento de la propagación y reutilización de funciones y la reducción notable de la cantidad de parámetros (Huang et al., 2018).

Los modelos de clasificación de los movimientos manuales generados al variar los parámetros de la CWT (figura 3.28) se entrenan con las características de la tabla 3.5 mediante la CNN *DenseNet*. En la tabla 3.8 se muestra la precisión, la pérdida de entropía cruzada (CEL), el tiempo de procesamiento, el tiempo de clasificación y el tiempo total de cada modelo de clasificación después del entrenamiento.

Tabla 3.8. Resultados del entrenamiento con la CNN *DenseNet*.

Función de Wavelet Madre		Número de escala	Precisión	CEL	Tiempo Proces. (ms)	Tiempo Clasif. (ms)	Tiempo total (ms)
Mexican Hat		2	0.8955	0.1053	12.95	189.99	202.94
		15	0.8867	0.3898	15.00	190.99	205.99
		30	0.9711	0.1193	17.99	188.48	206.47
		50	0.9289	0.0362	22.99	189.58	212.57
Morlet		2	0.7552	0.6496	12.99	192.99	205.98
		15	0.9184	0.2932	15.00	188.78	203.78
		30	0.9756	0.0387	18.00	189.03	207.03
		50	0.9422	0.1606	23.99	183.32	207.91
Derivada Gaussiana	Primer Orden	2	0.9543	0.1299	11.99	191.16	203.15
		15	0.9592	0.1638	14.54	186.00	200.54
		30	0.9543	0.2996	17.99	189.99	207.98
		50	0.9644	0.0708	22.05	189.05	211.10
	Segundo Orden	2	0.9244	0.0855	12.99	192.99	205.98
		15	0.9733	0.0544	15.99	189.00	204.99
		30	0.9667	0.0232	17.99	187.32	205.31
		50	0.9489	0.1878	22.00	194.99	216.99
	Tercer Orden	2	0.8667	0.4743	13.51	190.58	204.09
		15	0.9689	0.0381	15.99	192.99	208.98
		30	0.9778	0.0371	18.99	192.15	211.14
		50	0.8222	0.2211	21.85	191.76	213.61

Fuente: Elaboración propia

3.4.4. MobileNet V2

La arquitectura de la Red Neuronal *MobileNet* V2 tiene como enfoque la eficiencia del costo computacional para modelos de visión por computadora en sistemas con recursos limitados como las aplicaciones móviles y los sistemas embebidas. El objetivo de esta arquitectura es disminuir significativamente la cantidad de operaciones y la memoria necesaria mientras se mantiene una alta precisión de clasificación (Sandler et al., 2019).

La particularidad del *MobileNet* V2 es qué está formado por bloques de construcción llamados *“Inverted Residuals with Linear Bottlenecks”*, traducido en Residuales Invertidos con Cuellos de Botella Lineales (ver figura 3.33). El bloque consiste en tomar como entrada una representación de baja dimensión como píxeles para expandirlo a una alta dimensión cómo categorías de imágenes y filtrarlo con una convolución ligera, luego los datos se transforman nuevamente una a dimensión baja mediante una convolución lineal. Ello permite reducir la cantidad de memoria utilizada al no materializar completamente grandes tensores intermedios (Sandler et al., 2019).

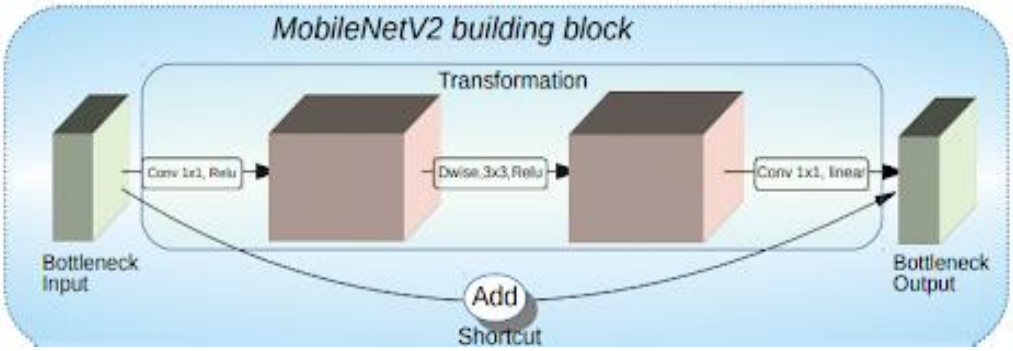


Figura 3.33. Bloque de construcción del *MobileNet* V2.
Fuente: Sandler et al. (2019) MobileNet V2 building block

Los modelos de clasificación de los movimientos manuales generados al variar los parámetros de la CWT (figura 3.28) se entrenan con las características de la tabla 3.5 mediante la CNN *MobileNet* V2. En la tabla 3.9 se muestra la precisión, la pérdida de entropía cruzada (CEL), el tiempo de procesamiento, el tiempo de clasificación y el tiempo total de cada modelo de clasificación después del entrenamiento.

Tabla 3.9. Resultados del entrenamiento con la CNN *MobileNet* V2.

Función de Wavelet Madre		Número de escala	Precisión	CEL	Tiempo Proces. (ms)	Tiempo Clasif. (ms)	Tiempo total (ms)
Mexican Hat		2	0.9511	0.2015	13.20	36.01	49.21
		15	0.9867	0.0031	15.00	34.25	49.25
		30	0.9600	0.0671	19.02	37.15	56.17
		50	0.9333	0.0202	27.00	35.98	62.98
Morlet		2	0.7178	1.0424	12.87	31.02	43.89
		15	0.9556	0.1403	15.00	34.03	49.03
		30	0.9467	0.3761	18.01	34.22	52.23
		50	0.9578	0.0325	22.72	34.54	57.26
Derivada Gaussiana	Primer Orden	2	0.9444	0.2942	13.03	34.96	47.99
		15	0.9644	0.0552	14.80	35.04	49.84
		30	0.9578	0.0611	18.02	34.97	52.99
		50	0.9133	0.4624	21.62	32.96	54.58
	Segundo Orden	2	0.8711	0.9110	12.54	32.97	45.51
		15	0.8867	0.7684	14.98	34.01	48.99
		30	0.9333	0.2506	17.02	32.97	49.99
		50	0.9333	0.1425	21.04	34.25	55.29
	Tercer Orden	2	0.7689	0.4874	12.96	36.00	48.96
		15	0.9733	0.0034	15.00	33.51	48.51
		30	0.9533	0.2781	17.34	33.02	50.36
		50	0.9444	0.1746	21.55	33.97	55.52

Fuente: Elaboración propia

3.4.5. Selección del modelo con mejor rendimiento

Para una correcta selección del modelo de clasificación se define 3 filtros de las métricas resultantes del entrenamiento de las 4 CNN, considerando el balance entre la precisión y la velocidad debido a que la latencia entre precisiones debe ser menor a 300 para tener un control eficiente en tiempo real (Englehart & Hudgins, 2003) (Nahid et al., 2020a). De este tiempo, la ventana deslizante emplea 200 ms. dejando 100 ms para los demás procesos.

El primer filtro planteado es una precisión mayor a 97%, el segundo filtro es una pérdida de entropía cruzada (CEL) menor a 0.1, y el último filtro es un tiempo total (tiempo de procesamiento más clasificación) menor a 100 ms. Los modelos con los mejores rendimientos que han pasado los 3 filtros se observan en la tabla 3.10.

Tabla 3.10. Modelos de clasificación con los mejores rendimientos.

CNN	Función de Wavelet Madre	Número de escala	Precisión	CEL	Tiempo total (ms)
AlexNet	Derivada de Primer Orden Gaussiana	50	0.9778	0.0857	35.98
MobileNet V2	*Mexican Hat*	15	0.9867	0.0031	49.25
MobileNet V2	Derivada de Tercer Orden Gaussiana	15	0.9733	0.0034	48.51

Fuente: Elaboración propia

De la tabla 3.6 hasta la tabla 3.10 se desprende las siguientes afirmaciones: La CNN *AlexNet* ha logrado tener un modelo que pase los 3 filtros, empero, sus métricas son inferiores a las obtenidas con la CNN *MobileNet* V2. La CNN *ResNet-18* no ha logrado tener ningún representante debido a que sus CEL eran mayores a 0.1. Por otra parte, la CNN *DenseNet* ha tenido buenas métricas; sin embargo, su tiempo total fue mayor a 200 ms.

Por los motivos antes mencionados, se seleccionó el modelo con la arquitectura *MobileNet* V2, función de Wavelet Madre *Mexican Hat* y número de escala 15, como el modelo de clasificación del proyecto. El modelo seleccionado resulta tener la mayor precisión y la menor pérdida de entropía cruzada (CEL); asimismo, su tiempo de procesamiento más clasificación es menor a 100 ms.

- **Características del modelo seleccionado**

El modelo del proyecto ha logrado un resultado de 97.52% y 98.67% de precisión de clasificación en el conjunto de entrenamiento y validación respectivamente; asimismo, ha obtenido pérdidas de entropía cruzada (CEL) del entrenamiento y la validación de 0.0809 y 0.0035 respectivamente. Los bajos valores de las diferencias de las precisiones (1.15 %) y de las CEL (0.0774) indican que no se ha producido un sobreajuste durante el entrenamiento. La figura 3.34 muestra el progreso las precisiones y las CEL del entrenamiento y validación durante las 3 etapas.

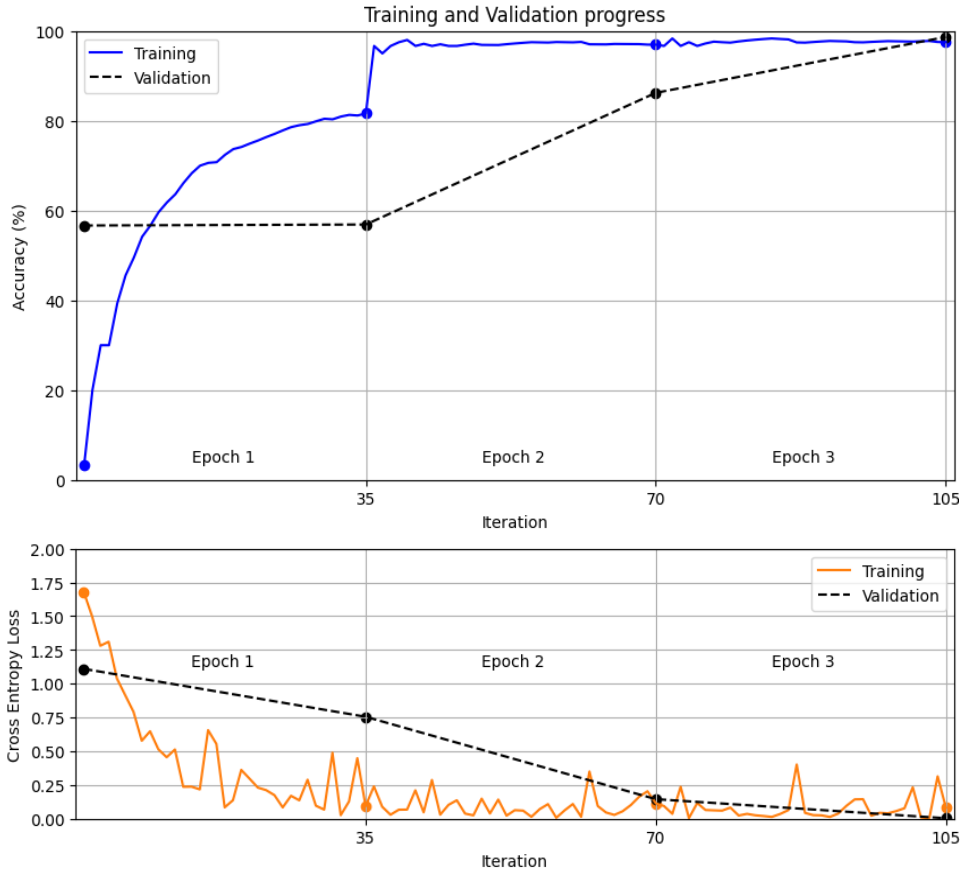


Figura 3.34. Gráfica del progreso del entrenamiento y validación del modelo seleccionado (Precisión vs. Iteración y CEL vs. Iteración).
Fuente: Elaboración propia

En la matriz de confusión de la figura 3.35 se muestra que 5 de los 6 errores de clasificación se producen entre el empuje de plataforma (PlPush) y la pinza pulgar-índice (ThIndex), el motivo podría ser el hecho de que en ambos movimientos mantienen la extensión de los dedos medio, anular y meñique.

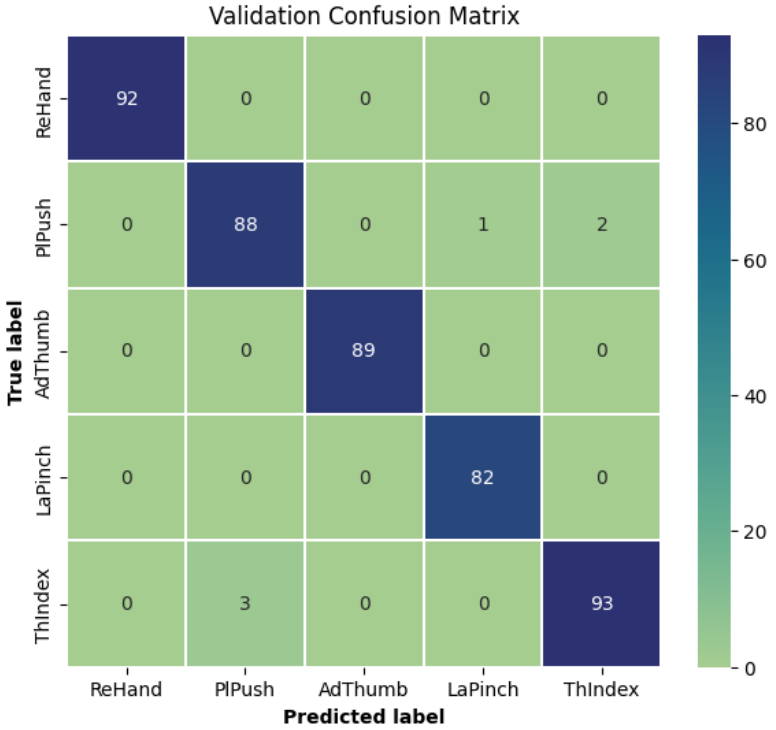


Figura 3.35. Matriz de confusión del modelo seleccionado.
Fuente: Elaboración propia

3.4.6. Conclusión del subsistema de clasificación

El subsistema de clasificación ha evaluado y comparado el entrenamiento de 4 arquitecturas de Deep Learning para la clasificación de los 5 movimientos manuales variando los parámetros de la CWT. A partir de ello, se concluyó con la selección del modelo con la arquitectura *MobileNet* V2, función de Wavelet Madre *Mexican Hat* y número de escala 15 debido a su rendimiento superior entre 80 diferentes modelos propuestos.

3.5. Diseño de la Interfaz Gráfica de Usuario de señales EMG

Con el fin de interactuar de manera eficiente con los subsistemas de adquisición, procesamiento y clasificación, logrando un control en tiempo real para una prótesis, se desarrollará una Interfaz gráfica de Usuario (GUI) de señales EMG para los usuarios considerando los principios de diseño de una GUI propuestos por (Martinez, 2011). Por ejemplo, uno de los principios siguiera la construcción de un modelo conceptual del flujo de los procesos que deben ocurren en la GUI para tener una programación eficiente.

Después del desarrollo de los principios, se programará mediante hilos de ejecución en paralelo: la adquisición, visualización, procesamiento, entrenamiento y clasificación de las señales empleando el lenguaje de programación Python. Se opta por una programación de hilos debido a que los procesos de adquisición y procesamiento señales biológicas en una GUI deben ejecutarse de forma paralela, de tal manera que el muestreo a mil Hertz de las señales EMG sea ininterrumpido (Sepúlveda et al., 2015)

3.5.1. Principios de diseño para la GUI

La GUI se diseñó a partir de los cuatro principios de diseño de una GUI de (Martínez, 2011). A continuación, se desarrolla cada principio siguiendo las recomendaciones del autor.

- **Enfócate en los usuarios y sus tareas, no en la tecnología**

  Para cumplir con el primer principio de diseño, (Martínez, 2011) siguiere responder las preguntas enunciadas a continuación:

  - ¿Quiénes son los usuarios previstos para la GUI?

    Las personas que necesiten de una prótesis de mano,

médicos, terapeutas, estudiantes, investigadores, entre otros. En general, personas con interés en el funcionamiento del sistema de control de una prótesis.

- ¿Qué actividades debe apoyar la GUI?

  La GUI debe permitir la adquisición de las señales, la visualización de las gráficas de los 3 canales, el procesamiento de las señales, el entrenamiento del modelo de clasificación, y la predicción en tiempo real cuando el usuario realice un movimiento manual.

- ¿Qué problemas debería resolver la GUI?

  El principal problema es la necesidad de la paralelización de los procesos de la adquisición a una frecuencia de muestreo de mil Hertz y la predicción del movimiento manual con una latencia menor a 300 ms para ser efectivo el control en tiempo real de una prótesis.

- ¿Cuáles son las tareas que el usuario debe realizar?

  Las tareas del usuario son: el control de la comunicación serial de la GUI con el subsistema de adquisición y la prótesis; la adquisición de las señales por cada movimiento manual durante 60 segundos; y el control del inicio del procesamiento de las señales, el entrenamiento del modelo de clasificación y la predicción del movimiento manual en tiempo real.

- **Considerar la función primero, la presentación después**

Para cumplir con este principio, (Martinez, 2011) siguiere construir un modelo conceptual para comprender los procesos que deben ocurrir en la GUI. En la figura 3.36 se diseñó el modelo conceptual que cumpla con las actividades que debe apoyar la GUI mediante hilos de ejecución. En el subcapítulo 3.5.2 se explicará el hilo principal del GUI; en el subcapítulo 3.5.3, el hilo de adquisición; en el capítulo 3.5.4, los hilos de las gráficas de los canales; y en el subcapítulo 3.55, el hilo de procesamiento y clasificación.

- **Conforme a la visión del usuario, y no lo complique más**

Según (Martínez, 2011), las GUI no deben incluir muchas animaciones llamativas y gráficos coloridos, así como controles de interfaz que sean innecesarios. Para cumplir con este principio, durante la programación se debe implementar solamente las funciones nombradas en el modelo conceptual de la GUI propuesto. Además, se debe utilizar pocos colores en el diseño gráfico.

- **Promover el aprendizaje y entregar información, no solo datos**

Según (Martínez, 2011), las GUI deben promover el aprendizaje rápido del funcionamiento. Por lo tanto, se debe brindar información útil como: las gráficas de las señales en tiempo real, el estado actual de los diferentes procesos, el porcentaje de precisión del modelo entrenado, el movimiento manual estimado, entre otros.

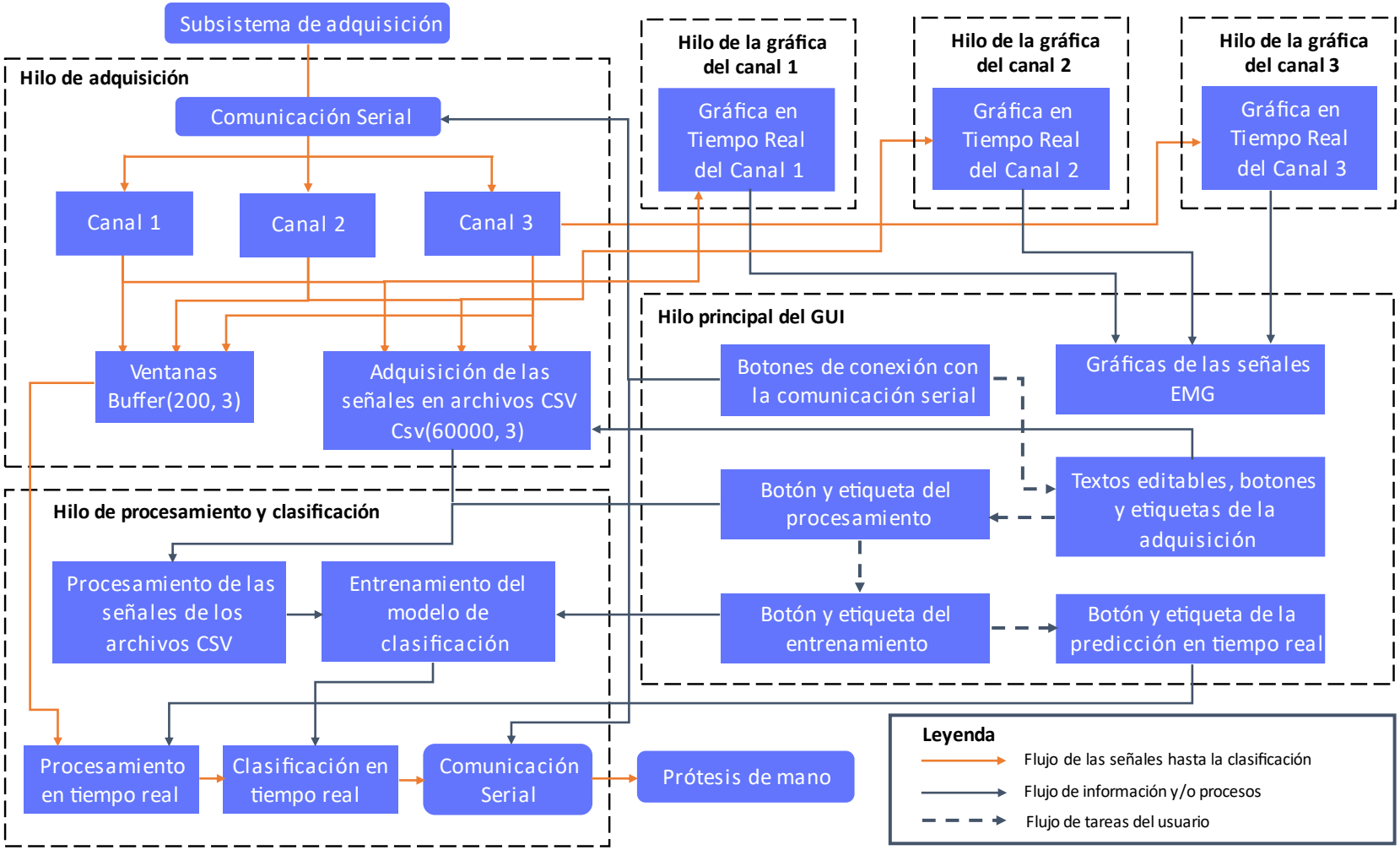


Figura 3.36. Modelo conceptual de la GUI propuesta.

Fuente: Elaboración propia

3.5.2. Programación del hilo principal de la GUI

En la figura 3.36 se muestra que el hilo principal de la GUI propuesta tiene como objetivo la programación de las etiquetas, textos editables, gráficas y botones, los cuales son los medios para recibir y enviar información con los otros hilos. Asimismo, es el único hilo de ejecución que permite al usuario interactuar y realizar las diferentes tareas que se plantearon en el desarrollo de los principios de diseño de (Martinez, 2011). El resultado final del diseño la programación de la GUI se muestra en la figura 3.37.

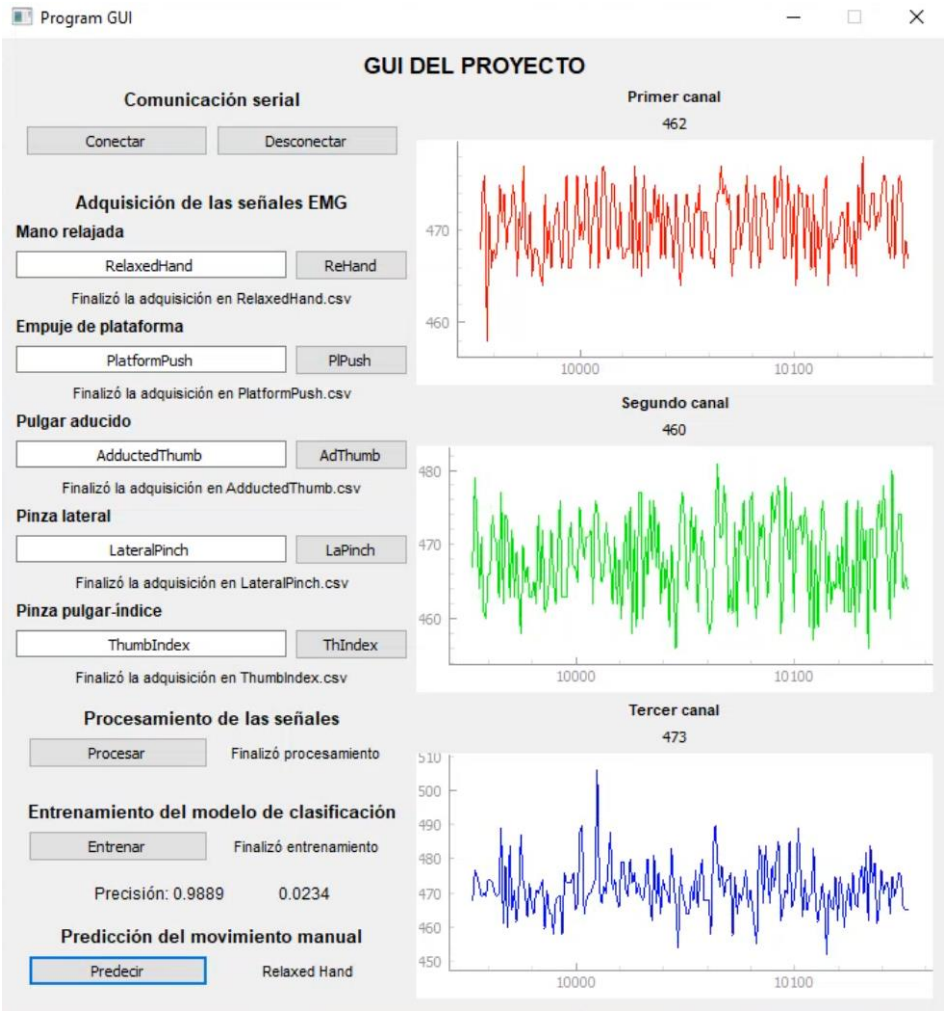


Figura 3.37. GUI del Proyecto.
Fuente: Elaboración propia

3.5.3. Programación del hilo de adquisición

El hilo de adquisición de las señales tiene como primer objetivo recibir las tres señales EMG a una frecuencia de 1000 Hz mediante comunicación serial con una velocidad de 115200 Baudios desde el subsistema de adquisición. Para ello, el control de la comunicación se desarrolló a través de los botones “Conectar” y “Desconectar” para iniciar e interrumpir la comunicación respectivamente (ver figura 3.38).

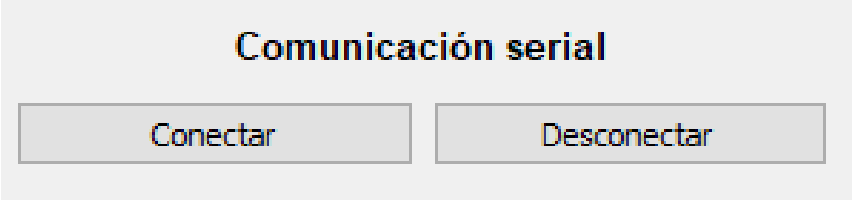


Figura 3.38. Botones de control de la comunicación serial.
Fuente: Elaboración propia

El segundo objetivo es ayudar al usuario a adquirir las señales EMG cuando se realice cada movimiento manual durante 60 segundos (60000 datos en total), para luego ser utilizadas en el entrenamiento del modelo de clasificación. Por ello, en la figura 3.39 se puede observar que se programó: textos editables para nombrar los archivos CSV donde se guardará los datos, botones que ayuden a controlar el inicio de la adquisición de las señales, y etiquetas que indican el estado de la adquisición de cada movimiento manual. Durante las pruebas de la GUI se ha obtenido que el tiempo total aproximado para adquirir las señales es de 6 minutos y 45 segundos.

El último objetivo del hilo de adquisición es generar contantemente las tres ventanas de 200 ms, las cuales serán procesadas y clasificadas en tiempo real en otro hilo de ejecución

paralelamente cuando el modelo de clasificación finalice su entrenamiento, tal como se observa en la figura 3.36.

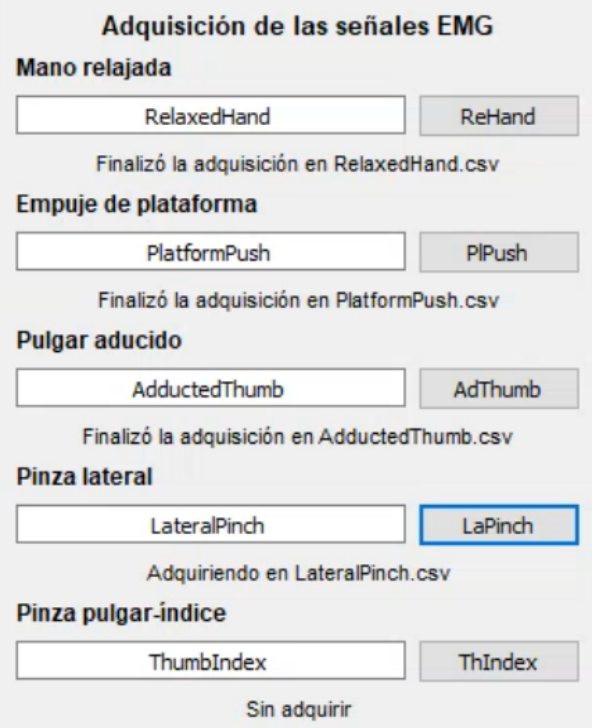


Figura 3.39. Parte de la GUI para la adquisición de las señales EMG.
Fuente: Elaboración propia

3.5.4. Programación de los hilos de las gráficas de los canales

Los hilos de las gráficas de los canales tienen como objetivo graficar las señales EMG recibidas en tiempo real desde el hilo de adquisición (ver figura 3.36) para que el usuario puede identificar rápidamente como las señales EMG varían en magnitud y frecuencia durante un movimiento manual. Si el cambio no sucede, posiblemente se debe a un mal posicionamiento de los electrodos.

Se programó un hilo por cada canal para poder mantener la fluidez de las gráficas y de la GUI en general. Las gráficas de las señales EMG en tiempo real se observan en la figura 3.40.

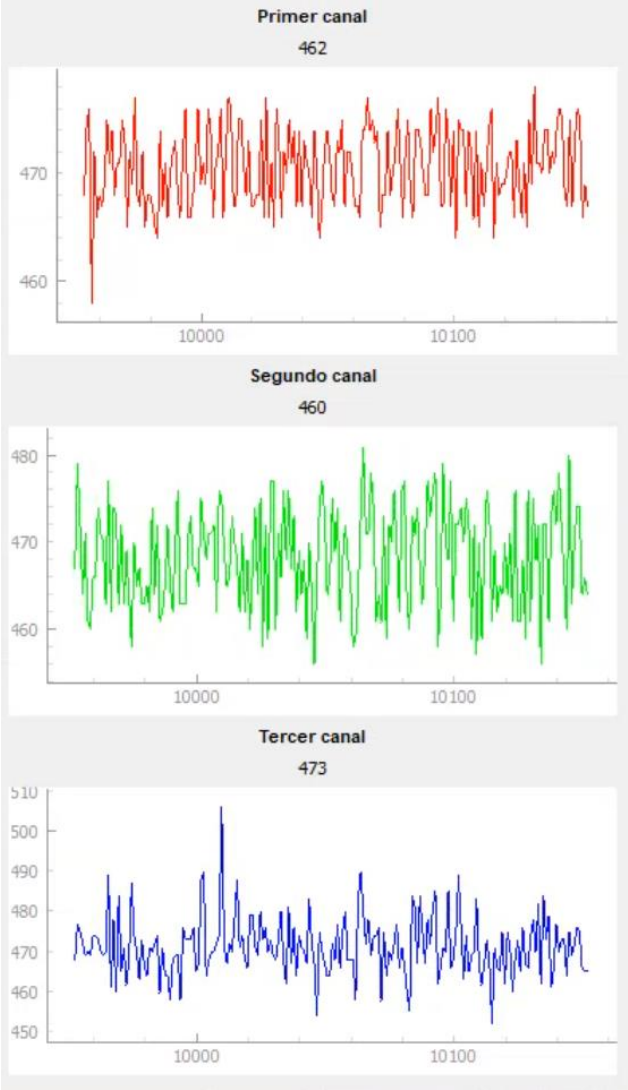


Figura 3.40. Gráficas de los 3 canales de los sensores EMG.
Fuente: Elaboración propia

3.5.5. Programación del hilo de procesamiento y clasificación

Según el modelo conceptual (ver figura 3.36), el hilo de procesamiento y clasificación tiene 3 procesos que no suceden en forma paralela siendo dependentes del resultado de su predecesor.

El primero proceso es el procesamiento de las señales de los archivos CSV generados en el hilo de adquisición, para obtener escalogramas en archivos JPG mediante la CWT con parámetros de wavelet madre *Mexican Hat* y valor de escala 15.

Durante las pruebas de la GUI se ha obtenido que este proceso toma un tiempo aproximado de 61 segundos. En figura 3.41 se observa que se desarrolló el botón “Procesar” para iniciar con el procesamiento de las señales y una etiqueta que indique el estado actual del procesamiento (“Sin procesar”, “Procesando” o “Finalizó procesamiento”).

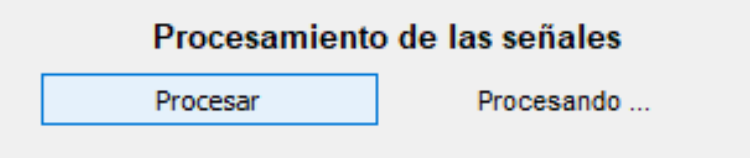


Figura 3.41. Parte de la GUI para el procesamiento.
Fuente: Elaboración propia

El segundo proceso es el entrenamiento del modelo de clasificación con la arquitectura *MobileNet* V2 teniendo como entrada los archivos JPG de los escalogramas. Durante las pruebas de la GUI se ha obtenido que este proceso toma un tiempo aproximado de 5 minutos y 24 segundos por cada etapa de entrenamiento.

En figura 3.42 se observa que se desarrolló el botón “Entrenar” para iniciar con el entrenamiento del modelo; una etiqueta que indique el estado actual del entrenamiento (“Sin entrenar”, “Entrenando” o “Finalizó entrenamiento”); y una etiqueta que indique la precisión y la pérdida de entropía cruzada (CEL) del modelo entrenado.

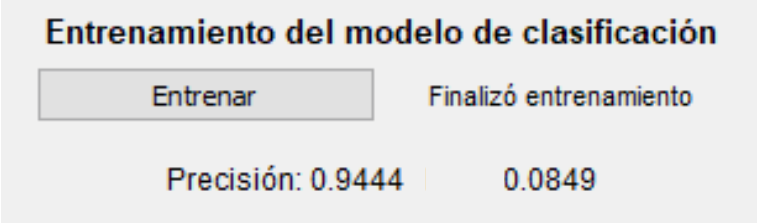


Figura 3.42. Parte de la GUI para el entrenamiento.
Fuente: Elaboración propia

El tercer proceso es la predicción del movimiento manual que el usuario este realizando con una latencia menor a 300 ms para el control en tiempo real de una prótesis. Para ello; en primer lugar, se recibe las 3 ventanas de datos EMG desde el hilo de adquisición; en segundo lugar, se procesa las ventanas hasta obtener 3 escalogramas; en tercer lugar, se clasifica en 1 de los 5 movimientos manuales mediante el modelo previamente entrenado; y, finalmente se envía el resultado de la clasificación mediante comunicación serial hacia la prótesis de mano tal como se observa en la figura 3.36.

Durante las pruebas de la GUI sea ha obtenido que este proceso toma un tiempo aproximado de 249.25 ms. En figura 3.43 se observa el botón “Predecir” para iniciar con la predicción del movimiento manual y una etiqueta con el nombre del movimiento manual que se está clasificando en tiempo real.

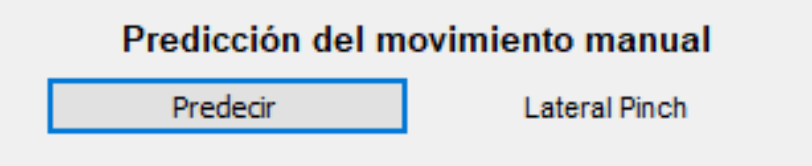


Figura 3.43. Parte de la GUI para la predicción en tiempo real.
Fuente: Elaboración propia

3.5.6. Conclusión de la GUI de señales EMG

La GUI ha tenido un correcto desarrolló debido al seguimiento de los principios de diseño de una GUI (Martinez, 2011). Por ejemplo, la construcción del modelo conceptual ha permitido una programación eficiente del flujo de la información y de las funciones que la GUI debe apoyar. Adicionalmente, debido a la programación en hilos de ejecución

en paralelo es posible adquirir las señales EMG con un muestro a mil Hertz de frecuencia y obtener una latencia menor de 300 ms para el control en tiempo real de una prótesis. Durante las pruebas se ha verificado que la GUI permite al usuario cumplir con todas tareas propuestas para obtener la predicción en tiempo real de los movimientos manuales en un tiempo total de 13 min y 10 s. En los ANEXO A.3 y ANEXO A.4 se muestran el código fuente de la GUI en Python a partir del modelo conceptual.

3.6. Simulación de una prótesis de mano

Según (Parajuli et al., 2019), un correcto control de una prótesis es posible mediante un sistema de retroalimentación visual, debido a la ayuda que brinda a los usuarios a familiarizarse y aprender sobre el sistema de control, significando una mayor precisión de clasificación.

Debido a que (Parajuli et al., 2019) también menciona que las prótesis virtuales pueden ser una alternativa de sistema de retroalimentación visual, se decide simular en el software Webots® el diseño de código abierto de la mano robótica antropomórfica ALARIS de 6 grados de libertad, el cual se puede obtener en la página web del laboratorio de los autores para una mayor personalización y utilización con fines de investigación (Nurpeissova et al., 2021).

En la figura 3.44 se observa como el desarrollo de la simulación de la mano ALIRIS seguirá una adaptación de la metodología del trabajo de (Forero et al., 2022) en tres etapas.

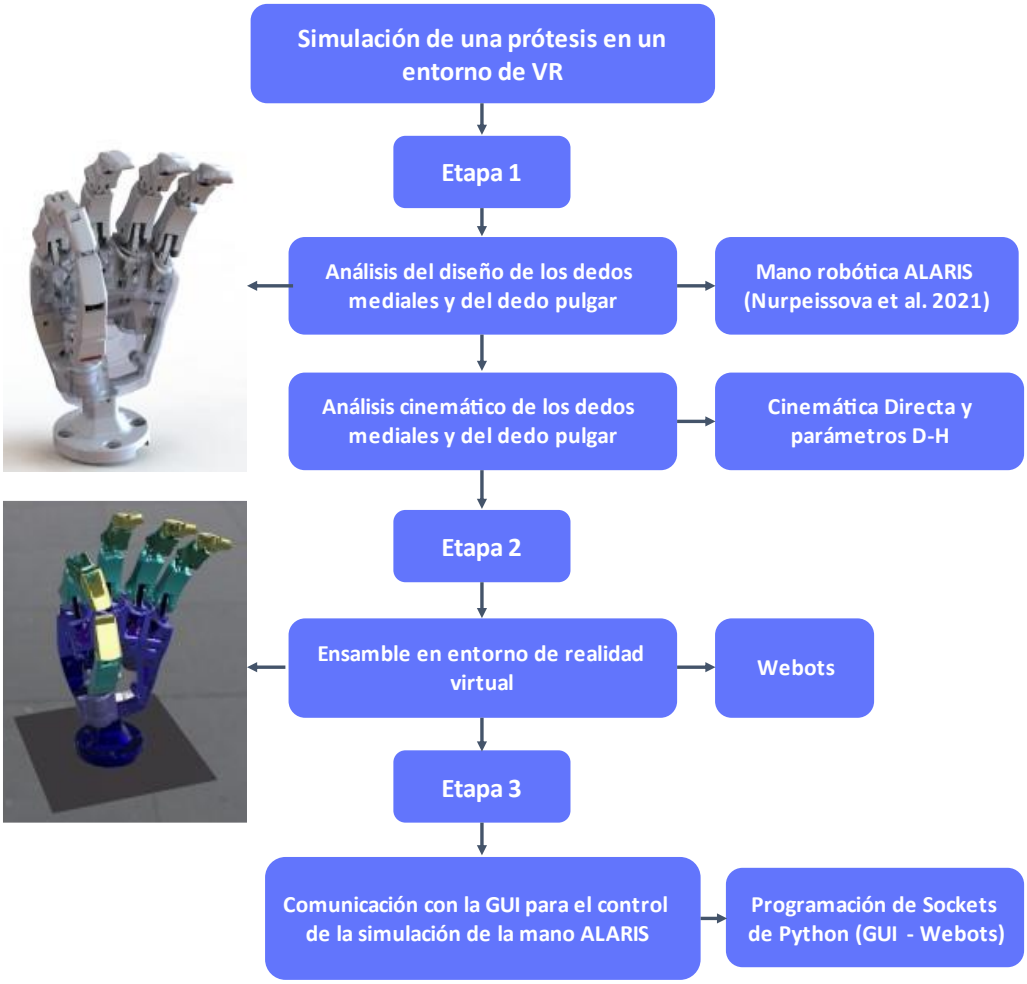


Figura 3.44. Metodología para la simulación de una prótesis.
Fuente: Elaboración propia en base del trabajo de Forero et al. (2022)

La primera etapa abarca el análisis del diseño de los dedos mediales y del dedo pulgar para comprender sus mecanismos de transmisión de movimiento; asimismo, el análisis cinemático de los dedos mediales y del dedo pulgar mediante la convección D-H. La segunda etapa consiste en el ensamblaje del diseño en un entorno virtual como Webots®, se importará el diseño de la mano ALARIS para luego configurar los árboles nodales que permitan simular los movimientos de las articulaciones y programar el controlador de los actuadores. La última etapa es la implementación de la comunicación la GUI con la simulación de la mano ALARIS mediante sockets

en una misma red local, para que un usuario, con los electrodos EMG conectados y utilizando la GUI, pueda tener una retroalimentación visual después de realizar un movimiento manual.

3.6.1. Análisis del diseño y la cinemática de la mano ALARIS

Durante la primera fase de la metodología del trabajo de (Forero et al., 2022) se diseñó y verificó el mecanismo de movimiento interfalángico para programar la cinemática y las trayectorias de los dedos; asimismo, se diseñó el modelo CAD de la prótesis para ensamblar las piezas en un entorno de VR en la segunda fase. Para cumplir los mismos objetivos de la primera fase, se realiza el análisis del diseño y la cinemática de los dedos mediales y del dedo pulgar de la mano robótica antropomórfica ALARIS de 6 grados (ver figura 3.45).

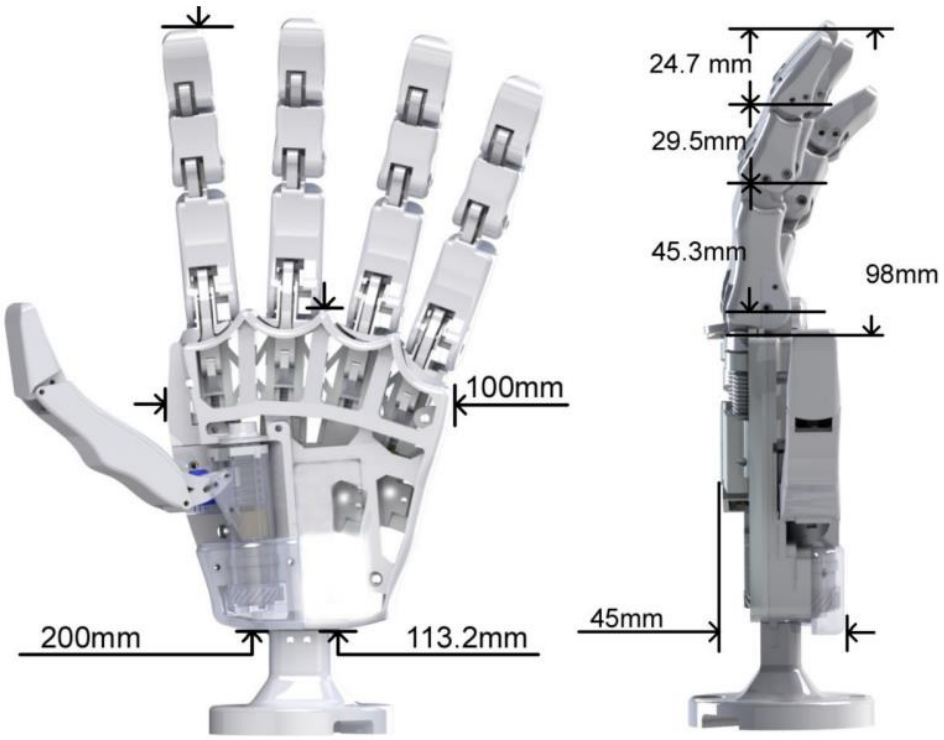


Figura 3.45. Diseño de la mano ALARIS con dimensiones.
Fuente: Nurpeissova et al. (2021) ALARIS hand design with dimensions

- **Análisis del diseño de los dedos mediales**

Los dedos mediales de la mano ALARIS tienen un único diseño y cada dedo tiene un grado de libertad que permite los movimientos de extensión y flexión mediante tres falanges: falange distal (DP), falange medio (MP), y la falange proximal (PP). El mecanismo que permite el movimiento de las articulaciones metacarpofalángica (MCP), interfalángica distal (DIP) e interfalángica proximal (PIP) es un sistema de enlace acoplados de cuatro barras (Nurpeissova et al., 2021). Este sistema se muestra en la figura 3.46.

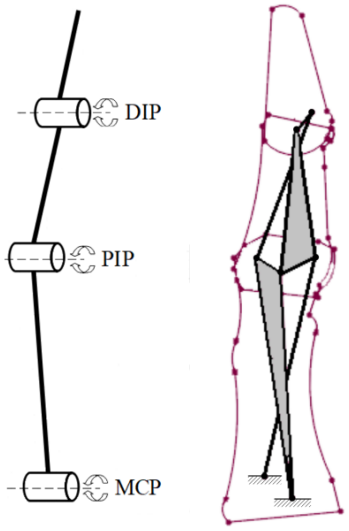


Figura 3.46. (a) Articulaciones de los dedos mediales (b) Sistema de enlaces acoplados de cuatro barras.
Fuente: Nurpeissova et al. (2021) (a) Hand finger joints (b) system of coupled four-bar linkages of the finger mechanism

La vista explosionada del diseño del dedo medial se observa en la figura 3.47, donde: (1) falange distal (DP); (2) falange media (MP); (3) enlace; (4) falange proximal (PP); (5) enlace; (6) enlace; (7) base del dedo; (8) cojinete; (9) cremallera; (10) tornillo sin fin; (11) motor DC; y (12) sensor de posición. El movimiento de rotación del eje del motor

conectado al tornillo sin fin se transforma en el movimiento de translación de la cremallera ubicada en un riel de la palma de la mano. La cremallera se conecta con un enlace mediante un eje libre al sistema de enlaces acoplado de cuatro barras en la falange proximal para realizar los movimiento de flexión y extensión del dedo medial (Nurpeissova et al., 2021).

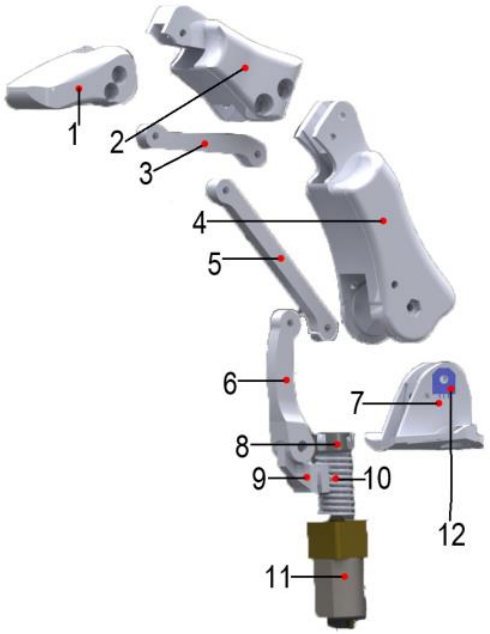


Figura 3.47. Vista explosionada del dedo medial.
Fuente: Nurpeissova et al. (2021) Exploded view of the hand finger.

- **Análisis del diseño del dedo pulgar**

El dedo pulgar de la mano ALARIS es un dedo adaptativo basado en el diseño de un dedo subactuado de dos falanges y dos grados de libertad de código abierto presentado en el trabajo de (Telegenov et al., 2015). El primer grado de libertad permite simular la abducción y aducción palmar en la articulación trapeciometacarpiana (TMC) mediante un mecanismo de engranajes helicoidales para producir una rotación axial del pulgar. El segundo grado de libertad

permite la flexión y extensión del pulgar a través del mecanismo de transmisión de tornillo sin fin y cremallera que acciona la articulación metacarpofalángica (MCP) y mediante un resorte en la falange proximal (PP) se transfiere el accionamiento a la articulación interfalángica (IP), tal como se muestra en la figura 3.48 (Nurpeissova et al., 2021).

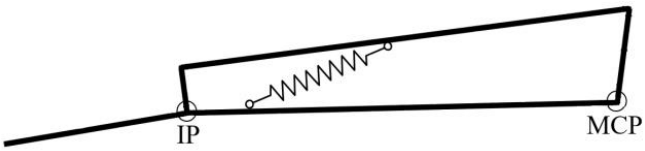


Figura 3.48. Modelo esquemático del pulgar de la mano subactuado.
Fuente: Nurpeissova et al. (2021) Schematic model of the underactuated hand thumb

La vista explosionada del diseño del dedo pulgar se observa en la figura 3.49, donde: (1) falange distal (DP); (2) falange proximal (PP); (3) enlace; (4) enlace; (5) sensor de posición; (6) cremallera; (7) tornillo sin fin; (8) motores DC; y (9) engranajes helicoidales de la rotación axial del pulgar (Nurpeissova et al., 2021).

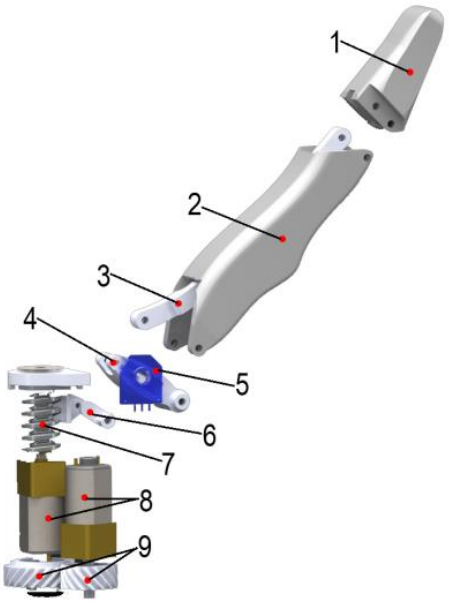


Figura 3.49. Vista explosionado del pulgar.
Fuente: Nurpeissova et al. (2021) Exploded view of the hand thumb

- **Análisis cinemático de los dedos mediales**

Se realiza el análisis cinemático del dedo medial de la mano ALARIS a través del empleo de la convección D-H (Denavit & Hartenberg, 1955) y el método de cinemática directa para describir el posicionamiento de las partes del dedo medial a partir de los movimientos angulares de las articulaciones.

Para poder empezar con el análisis, se realizó el dimensionado de las medidas de los enlaces que conforman el mecanismo de transmisión de movimiento, posteriormente se fijó los sistemas de coordenadas en las articulaciones principales del dedo medial (puntos O, B, E y G) tal como se observa en la figura 3.50, en el cual también está dibujado el sistema de enlaces acoplados de 4 barras. La primera barra es S_1, la segunda barra es el triángulo OBD, la tercera barra es S_2, y la cuarta barra es el triángulo BCE.

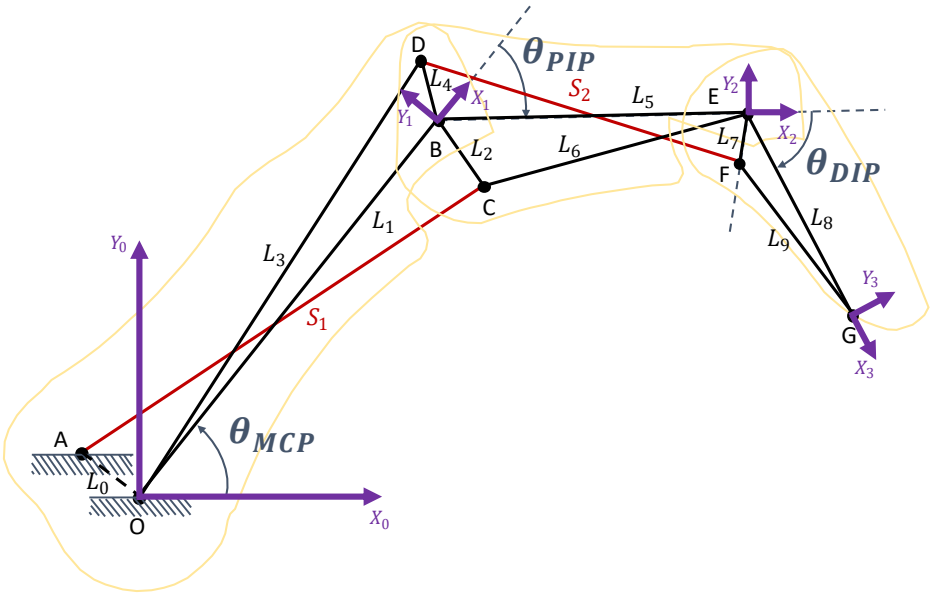


Figura 3.50. Sistemas de coordenadas del dedo medial.
Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

Las medidas de los enlaces que conforman el mecanismo de transmisión de movimiento del dedo medial se calcularon utilizando el software SolidWorks® y los archivos de libre acceso del trabajo de (Nurpeissova et al., 2021). Los resultados del dimensionado se muestran en la tabla 3.11.

Tabla 3.11. Medidas de los enlaces del mecanismo del dedo medial.

Enlace	Valor (mm)	Enlace	Valor (mm)
L_0	6.92	L_6	25.96
L_1	45.56	L_7	4.79
L_2	7.46	L_8	21.9
L_3	49.19	L_9	18.32
L_4	5.92	S_1	45.67
L_5	29.25	S_2	31.66

Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

A partir de los sistemas de coordenadas de la figura 3.50, se definió en la tabla 3.12 los parámetros Denavit-Hartenberg según los eslabones del dedo medial.

Tabla 3.12. Parámetros Denavit-Hartenberg del dedo medial.

Eslabones	θ_i	d_i	a_i	α_i
1	θ_{MCP}	0	L_1	0
2	θ_{PIP}	0	L_5	0
3	θ_{DIP}	0	L_8	0

Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

Con los parámetros definidos, se obtuvo la matriz homogénea final mencionada en la ecuación 2.5 del capítulo 2.1.14, al realizar la multiplicación las matrices de los 3 eslabones se obtuvo la matriz de la ecuación 3.7.

$$A_3^0(q) = \begin{bmatrix} \mathrm{C}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} & -\mathrm{S}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} & 0 & X_3 \\ \mathrm{S}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} & \mathrm{C}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} & 0 & \mathrm{Y}_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.7)$$

Donde (X_3, Y_3) es la posición del sistema coordenado final del dedo. En la ecuación 3.8 y 3.9 se tiene el valor de X_3 y Y_3 respectivamente.

$$X_3 = L_1 \mathrm{C}_{\theta_{MCP}} + L_5 \mathrm{C}_{(\theta_{MCP}+\theta_{PIP})} + L_8 \mathrm{C}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} \quad (3.8)$$

$$Y_3 = L_1 \mathrm{S}_{\theta_{MCP}} + L_5 \mathrm{S}_{(\theta_{MCP}+\theta_{PIP})} + L_8 \mathrm{S}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} \quad (3.9)$$

Debido a que la transmisión de movimiento en el dedo medial se realiza con el sistema de enlaces acoplados, los ángulos θ_{PIP}, θ_{DIP} y los ángulos de rotación de las barras S_1 y S_2 (θ_1 y θ_2 respectivamente) están en función de transmisión mecánica del ángulo θ_{MCP}. Se calcularon estás funciones matemáticamente para luego ser utilizadas en la simulación debido a que en el entorno virtual de Webots® el movimiento rotacional solo se realiza de un enlace respecto a otro, mas no respecto a dos o más enlaces.

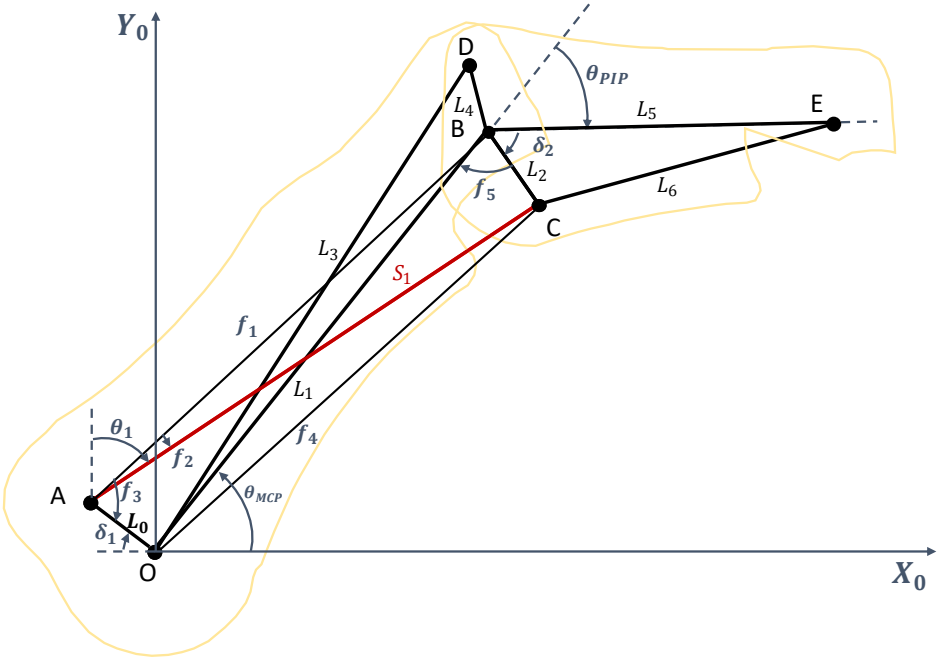


Figura 3.51. Geometría analítica de las falanges proximal y medial.
Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

La geometría análitica de las falanges proximal y medial de la mano ALARIS se muestra en la figura 3.51, dónde las funciones $f_1 - f_5$ sirven de intermediarios para calcular los valores de θ_1 y θ_{PIP} en función a θ_{MCP}.

La función f_1 es la longitud de AB y calculando por ley de cosenos en el triángulo OBA, se tiene la ecuación 3.10, donde δ_1 es un ángulo con el valor fijo de 37.77°.

$$f_1 = \sqrt{(L_0)^2 + (L_1)^2 - 2L_0L_1\cos(180° - \theta_{MCP} - \delta_1)} \quad (3.10)$$

La función f_2 es el ángulo $\measuredangle BAC$ y calculando por ley de cosenos en el triángulo CBA, se tiene la ecuación 3.11:

$$f_2 = arccos\left[\frac{(f_1)^2 + (S_1)^2 - (L_2)^2}{2f_1S_1}\right] \quad (3.11)$$

El ángulo θ_1 es el ángulo que forma la barra S_1 y el eje Y_0. Calculando θ_1 en el punto A, se tiene la ecuación 3.12:

$$\boxed{\theta_1 = 90° + \delta_1 + f_2 - f_3} \quad (3.12)$$

La función f_3 es el ángulo $\measuredangle OAB$ y calculando por ley de cosenos en el triángulo OAB, se tiene la ecuación 3.13:

$$f_3 = arccos\left[\frac{(L_0)^2 + (f_1)^2 - (L_1)^2}{2L_0f_1}\right] \quad (3.13)$$

La función f_4 es la longitud de OC y calculando por ley de cosenos en el triángulo OAC, se tiene la ecuación 3.14:

$$f_4 = \sqrt{(L_0)^2 + (S_1)^2 - 2L_0S_1\cos(f_3 - f_2)} \quad (3.14)$$

La función f_5 es el ángulo $\measuredangle CBO$ y calculando por ley de cosenos en el triángulo CBO, se tiene la ecuación 3.15:

$$f_5 = arccos\left[\frac{(S_1)^2 + (L_2)^2 - (f_4)^2}{2S_1L_2}\right] \qquad (3.15)$$

Calculando el ángulo θ_{PIP} en el punto B, se tiene la ecuación 3.16, donde δ_2 es un ángulo con el valor fijo de 57.06°.

$$\boxed{\theta_{PIP} = 180° - \delta_2 - f_5} \qquad (3.16)$$

En la figura 3.52 se observa la geometría análitica de las falanges medial y distal de la mano ALARIS, dónde las funciones $f_6 - f_{10}$ sirven de intermediarios para calcular los valores de θ_2 y θ_{DIP} en función a θ_{MCP}.

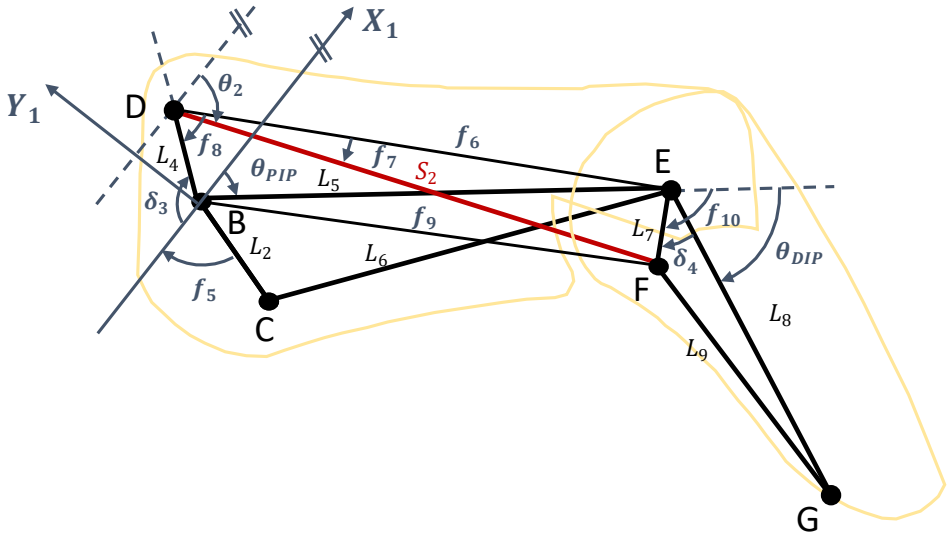


Figura 3.52. Geometría analítica de las falanges medial y distal.
Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

La función f_6 es la longitud de DE y calculando por ley de cosenos en el triángulo BDE, se tiene la ecuación 3.17, donde δ_3 es un ángulo con el valor fijo de 124.93°.

$$f_6 = \sqrt{(L_4)^2 + (L_5)^2 - 2L_4L_5\cos(\theta_{PIP} + 180° - \delta_3)} \qquad (3.17)$$

La función f_7 es el ángulo $\measuredangle FDE$ y calculando por ley de cosenos en el triángulo DEF, se tiene la ecuación 3.18:

$$f_7 = arccos\left[\frac{(f_6)^2 + (S_2)^2 - (L_7)^2}{2f_6S_2}\right] \qquad (3.18)$$

La función f_8 es el ángulo $\measuredangle EBD$ y calculando por ley de cosenos en el triángulo BDE, se tiene la ecuación 3.19:

$$f_8 = arccos\left[\frac{(f_6)^2 + (L_4)^2 - (L_5)^2}{2f_6L_4}\right] \qquad (3.19)$$

El ángulo θ_2 es el ángulo que forma la barra S_2 y el eje X_1. Calculando θ_1 en el punto D, se tiene la ecuación 3.20:

$$\boxed{\theta_2 = \delta_3 + f_7 - f_8} \qquad (3.20)$$

La función f_9 es la longitud de BF y calculando por ley de cosenos en el triángulo BDF, se tiene la ecuación 3.21:

$$f_9 = \sqrt{(L_4)^2 + (S_2)^2 - 2L_4S_2\cos(f_8 - f_7)} \qquad (3.21)$$

La función f_{10} es la longitud de $\measuredangle BEF$ y calculando por ley de cosenos en el triángulo BEF, se tiene la ecuación 3.22:

$$f_{10} = 180° - arccos\left[\frac{(L_5)^2 + (L_7)^2 - (f_9)^2}{2L_5L_7}\right] \qquad (3.22)$$

Calculando el ángulo θ_{DIP} en el punto E, se tiene la ecuación 3.23, donde δ_4 es un ángulo con el valor fijo de 37.28°.

$$\boxed{\theta_{DIP} = f_{10} - \delta_4} \qquad (3.23)$$

Una vez obtenido los valores de los ángulos θ_{PIP}, θ_{DIP}, θ_1 y θ_2; se pudo obtener el rango de movimiento del efector final del dedo medial y los valores de los sistemas de coordenados a partir del ángulo θ_{MCP}, tal como se muestra en la figura 3.53.

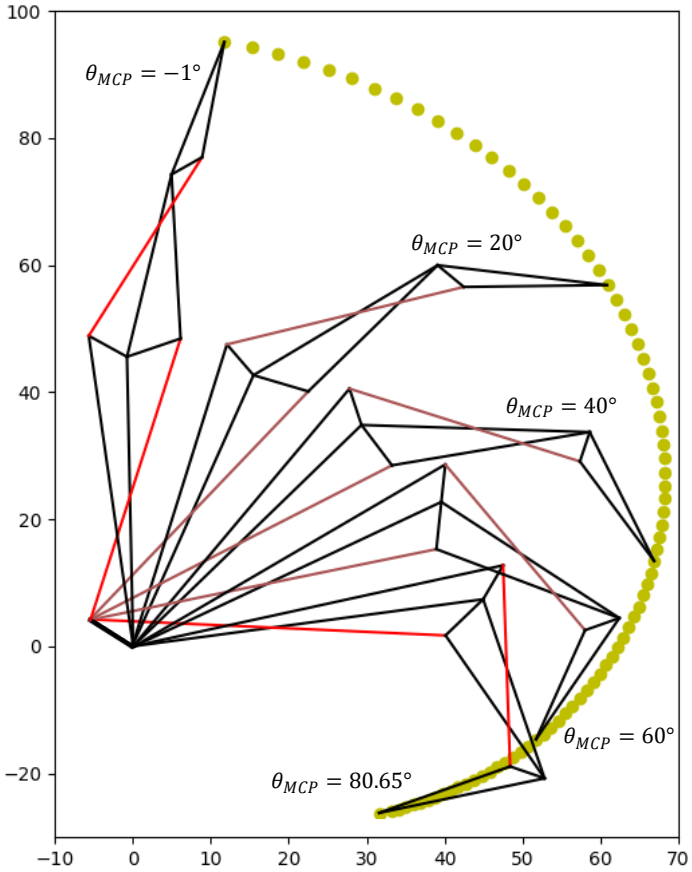


Figura 3.53. Rango de movimiento del dedo medial.
Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

- **Análisis cinemático del dedo pulgar**

De la misma manera que se realizó el análisis cinemático de los dedos mediales, se empleó la convección D-H y el método de cinemática directa para describir las posiciones de las partes del dedo pulgar a partir del movimiento de las articulaciones.

Se realizó el dimensionado de medidas de los enlaces que conforman el mecanismo de transmisión de movimiento, posteriormente se fijó los sistemas de coordenadas en las articulaciones principales del dedo pulgar (puntos O, A, B, E, F y H) tal como se observa en la figura 3.54, en el cual también está dibujado el pulgar subactuado de dos falanges. El primer grado de libertad es la

rotación del ángulo θ_{TMC} en el eje Z_0. El segundo grado de libertad para producir el movimiento de la articulación MCP y IP, empieza su accionamiento en el movimiento vertical en el eje Y_1 con el valor de X_{MCP}. Debido a que el enlace L_4 debe pasar en todo momento por el punto D, la variación del valor de X_{MCP} permite que el enlace L_4 gire con un ángulo de valor θ_{MCP_1} y la falange L_5 gire con un ángulo de valor θ_{MCP_2}. Ambos giros y la elongación del resorte del dedo pulgar permiten el giro con ángulo θ_{IP} de la segunda falange (triángulo GFH) de manera subactuada. Con el fin de simplificar los cálculos, se representa la longitud del resorte con el lado DF de valor R.

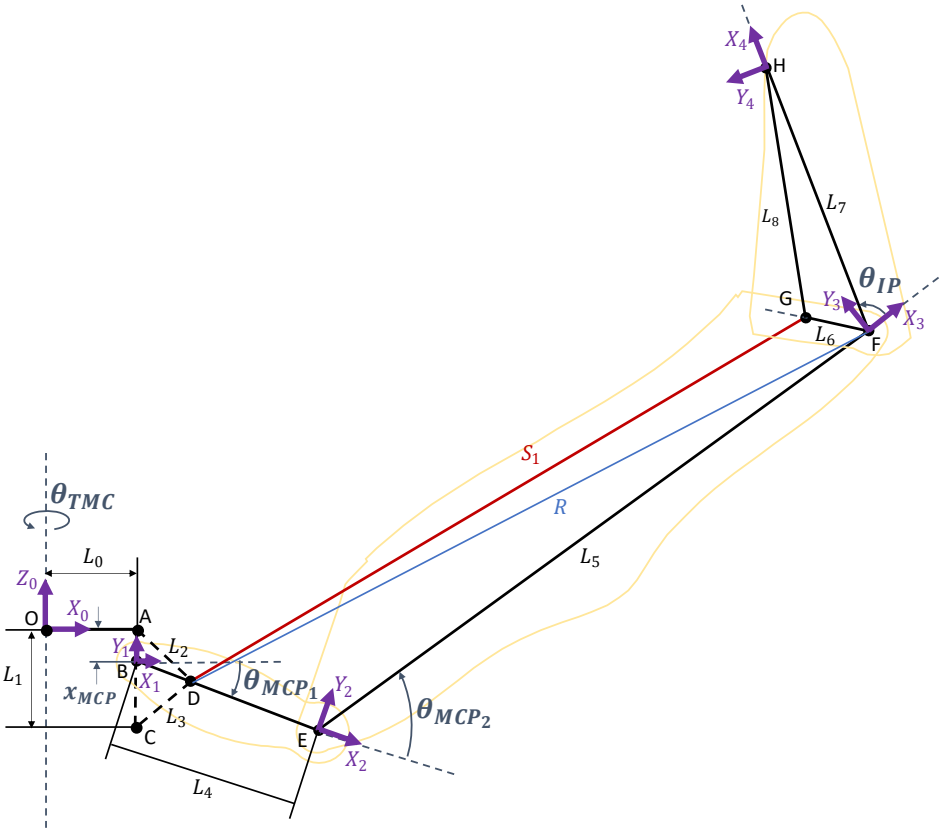


Figura 3.54. Sistemas de coordenadas del dedo pulgar.
Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

Las medidas de los enlaces que conforman el mecanismo de transmisión de movimiento del dedo pulgar se calcularon utilizando el software SolidWorks® y los archivos de libre acceso del trabajo de (Nurpeissova et al., 2021). En la tabla 3.13 se menciona el resultado del dimensionado, donde el enlace R tiene un rango de valores permitidos mecánicamente el cual depende en gran medida del esfuerzo exterior aplicado en las falanges.

Tabla 3.13. Medidas de los enlaces del mecanismo del dedo pulgar.

Enlace	Valor (mm)	Enlace	Valor (mm)
L_0	9.93	L_6	6.26
L_1	9.4	L_7	25.54
L_2	7.35	L_8	24.93
L_3	6.9	S_1	70.5
L_4	19.22	R	71.5-73.7
L_5	67		

Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

A partir de los sistemas de coordenadas de la figura 3.54, se definió en la tabla 3.14 los parámetros Denavit-Hartenberg según los eslabones del dedo pulgar.

Tabla 3.14. Parámetros Denavit-Hartenberg del dedo pulgar.

Eslabones	θ_i	d_i	a_i	α_i
1	θ_{TMC}	$-x_{MCP}$	L_0	90°
2	θ_{MCP_1}	0	L_4	0
3	θ_{MCP_2}	0	L_5	0
4	θ_{IP}	0	L_7	0

Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

Con los parámetros definidos, se obtuvo la matriz homogénea final mencionada en el capítulo 2.1.1, al realizar la multiplicación las matrices de los 3 eslabones como se observa en la ecuación 3.24.

$$A_4^0(q) = \begin{bmatrix} \mathrm{C}_{\theta_{TMC}}\mathrm{C}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} & -\mathrm{C}_{\theta_{TMC}}\mathrm{S}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} & \mathrm{S}_{\theta_{TMC}} & X_4 \\ \mathrm{S}_{\theta_{TMC}}\mathrm{C}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} & \mathrm{S}_{\theta_{TMC}}\mathrm{S}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} & -\mathrm{C}_{\theta_{TMC}} & \mathrm{Y}_4 \\ \mathrm{S}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} & \mathrm{C}_{(\theta_{MCP}+\theta_{PIP}+\theta_{DIP})} & 0 & Z_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.24)$$

Donde (X_4, X_4, Z_4) es la posición del sistema coordenado final del dedo. En las ecuaciones 3.25, 3.26 y 3.27 se tiene el valor de X_4, X_4 y Z_4 respectivamente.

$$X_4 = \mathrm{C}_{\theta_{TMC}}\left(L_0 + L_4\mathrm{C}_{\theta_{MCP_1}} + L_5\mathrm{C}_{(\theta_{MCP_1}+\theta_{MCP_2})} + L_7\mathrm{C}_{(\theta_{MCP_1}+\theta_{MCP_2}+\theta_{IP})}\right) \quad (3.25)$$

$$Y_4 = \mathrm{S}_{\theta_{TMC}}\left(L_0 + L_4\mathrm{C}_{\theta_{MCP_1}} + L_5\mathrm{C}_{(\theta_{MCP_1}+\theta_{MCP_2})} + L_7\mathrm{C}_{(\theta_{MCP_1}+\theta_{MCP_2}+\theta_{IP})}\right) \quad (3.26)$$

$$Z_4 = L_4\mathrm{S}_{\theta_{MCP_1}} + L_5\mathrm{S}_{(\theta_{MCP_1}+\theta_{MCP_2})} + L_7\mathrm{S}_{(\theta_{MCP_1}+\theta_{MCP_2}+\theta_{IP})} - x_{MCP} \quad (3.27)$$

El movimiento del ángulo θ_{TMC} es independiente de los otros ángulos debido a su grado de libertad propio. Sin embargo, como consecuencia del mecanismo del dedo subactuado de dos falanges, los ángulos θ_{MCP_1}, θ_{MCP_2}, θ_{IP} y θ_1 están en función de transmisión mecánica de las longitudes x_{MCP} y R. Estás funciones se calcularon matemáticamente para luego ser utilizadas en la simulación.

En la figura 3.55 se observa la geometría análitica del dedo pulgar de la mano ALARIS, dónde las funciones $f_1 - f_4$ sirven de intermediarios para calcular los valores de θ_{MCP_1}, θ_{MCP_2}, θ_{IP} y θ_1 en función a los valores de x_{MCP} y R.

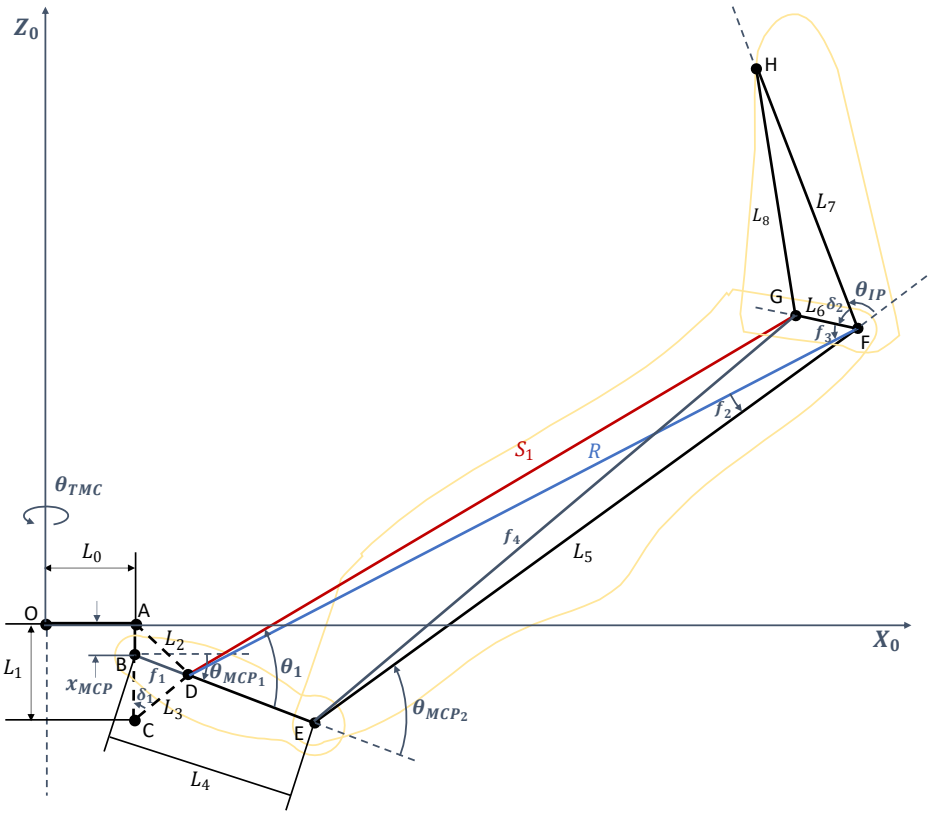


Figura 3.55. Geometría analítica del dedo pulgar de la mano ALARIS.
Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

La función f_1 es la longitud de BD y calculando por ley de cosenos en el triángulo BCD, se tiene la ecuación 3.28, donde δ_1 es un ángulo con el valor fijo de 50.82°.

$$f_1 = \sqrt{(L_1 - x_{MCP})^2 + (L_3)^2 - 2(L_1 - x_{MCP})L_3 \cos(\delta_1)} \quad (3.28)$$

Calculando θ_{MCP_1} (ángulo que forma el eje X_0 y la falange L_4) por ley de cosenos en el triángulo BDA, se tiene la ecuación 3.29:

$$\theta_{MCP_1} = 90° - arccos\left[\frac{(x_{MCP})^2 + (f_1)^2 - (L_2)^2}{2x_{MCP}f_1}\right] \quad (3.29)$$

Calculando θ_{MCP_2} (ángulo que forma las falanges L_4 y L_5) por ley de cosenos en el triángulo DEF, se tiene la ecuación 3.30:

$$\theta_{MCP_2} = 180° - arccos\left[\frac{(L_4 - f_1)^2 + (L_5)^2 - (R)^2}{2(L_4 - f_1)L_5}\right] \quad (3.30)$$

La función f_2 es el ángulo $\measuredangle EFD$ y calculando por ley de cosenos en el triángulo EFD, se tiene la ecuación 3.31:

$$f_2 = arccos\left[\frac{(L_5)^2 + (R)^2 - (L_4 - f_1)^2}{2L_5R}\right] \qquad (3.31)$$

La función f_3 es el ángulo $\measuredangle DFG$ y calculando por ley de cosenos en el triángulo DFG, se tiene la ecuación 3.32:

$$f_3 = arccos\left[\frac{(R)^2 + (L_6)^2 - (S_1)^2}{2RL_6}\right] \qquad (3.32)$$

Calculando el ángulo θ_{IP} en el punto F, se tiene la ecuación 3.33, donde δ_2 es un ángulo con el valor fijo de 77.36°.

$$\boxed{\theta_{IP} = 180° - \delta_2 - f_2 - f_3} \qquad (3.33)$$

La función f_4 es la longitud de EG y calculando por ley de cosenos en el triángulo EFG, se tiene la ecuación 3.34:

$$f_4 = \sqrt{(L_5)^2 + (L_6)^2 - 2L_5L_6\cos(f_2 + f_3)} \qquad (3.34)$$

Calculando θ_1 (ángulo que forma la falange L_4 y la falange S_1) por ley de cosenos en el triángulo DFG, se tiene la ecuación 3.35:

$$\boxed{\theta_1 = arccos\left[\frac{(L_4 - f_1)^2 + (S_1)^2 - (f_4)^2}{2(L_4 - f_1)S_1}\right]} \qquad (3.35)$$

Una vez obtenido los valores de los ángulos θ_{PIP}, θ_{MCP_1}, θ_{MCP_2} y θ_1; se obtuvo el rango de movimiento del efector final del dedo pulgar sin variar el ángulo θ_{TMC}, tal como se muestra en la figura 3.56, dónde el posicionamiento del dedo pulgar varía respecto a los valores de x_{MCP} y R.

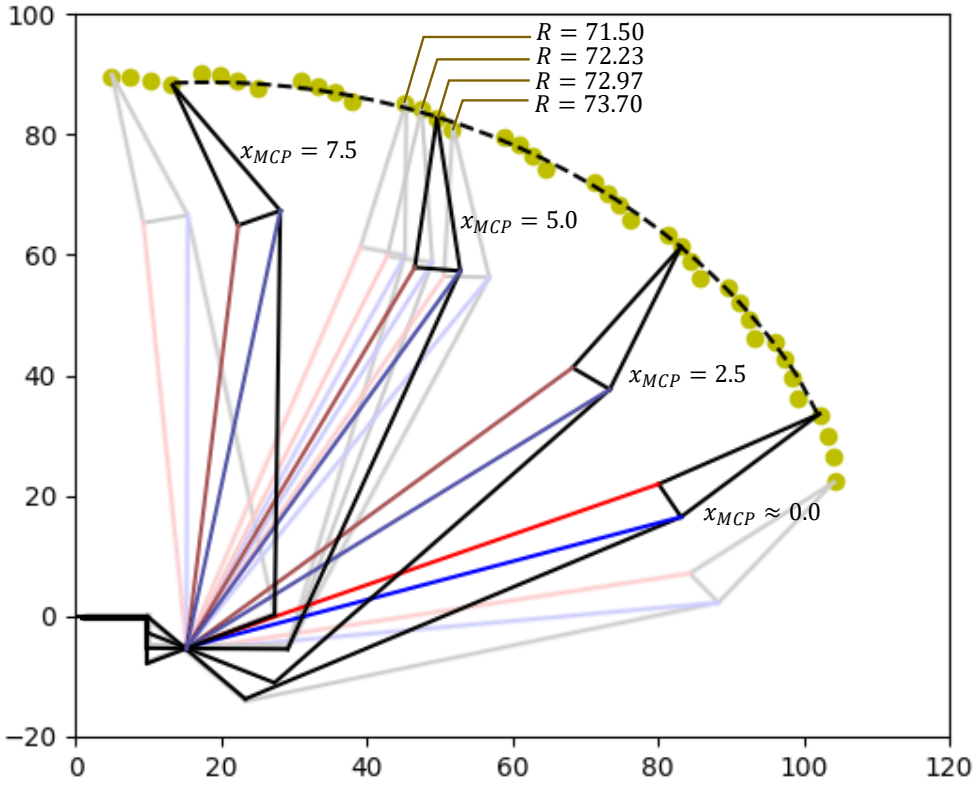


Figura 3.56. Rango de movimiento del dedo pulgar.
Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

Para la simulación del dedo pulgar se decide realizar una relación para que el valor de θ_{IP} varié proporcionalmente al valor θ_{MCP_1}. Esta relación está definida matemáticamente en la ecuación 3.37, y se sustenta en que las articulaciones MCP e IP dependen mayormente de los valores de x_{MCP} y R respectivamente.

$$R = \frac{R_{máx} - R_{mín}}{x_{MCP_{máx}} - x_{MCP_{mín}}} . x_{MCP} + R_{mín} \qquad (3.37)$$

Finalmente, se observa en la figura 3.57 la superficie que forma el rango de movimiento del efector final del dedo pulgar variando los dos grados de libertad teniendo como restricción la relación matemática de la ecuación 3.37.

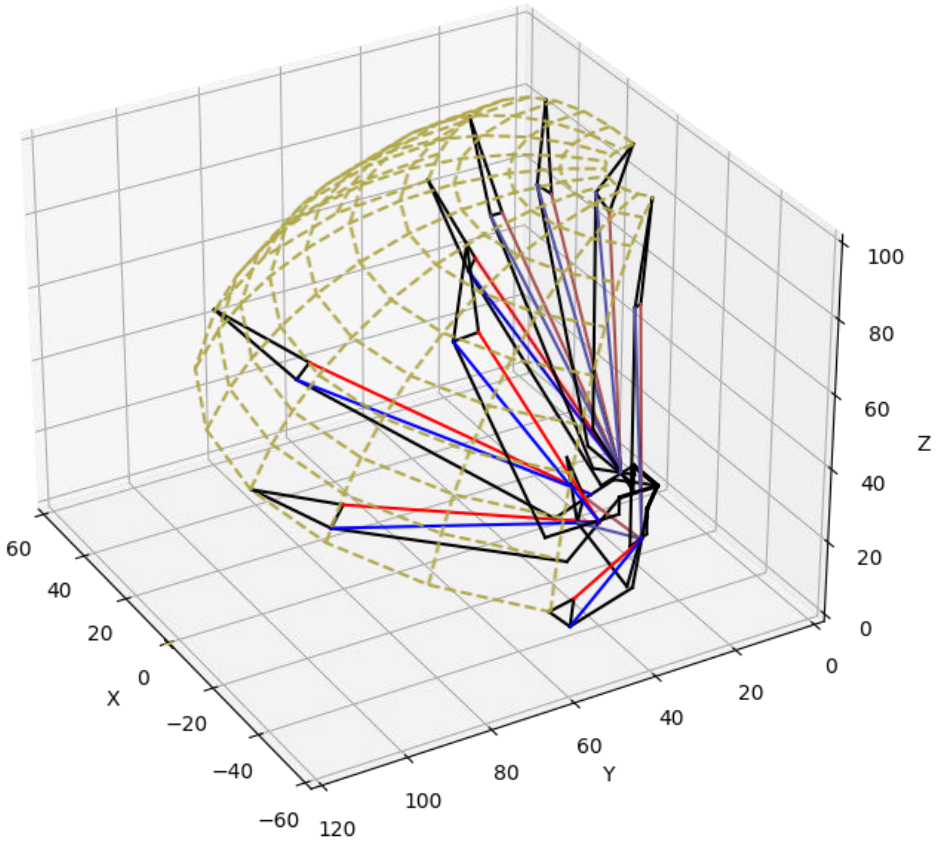


Figura 3.57. Superficie del rango de movimiento del dedo pulgar.
Fuente: Elaboración propia en base del trabajo de Nurpeissova et al. (2021)

3.6.2. Ensamblaje en el entorno de VR

La segunda fase de la metodología tiene como objetivo ensamblar la mano ALARIS en el software Webots® y programar la lógica de control que permita simular los movimientos manuales seleccionado. Para ello en primer lugar, se importó el diseño de la mano ALARIS en Webots®. En segundo lugar, se estableció los árboles nodales que permitan simular el movimiento de las articulaciones. En tercer lugar, se programó el control de los actuadores teniendo en cuenta las funciones de transmisión mecánicas calculadas en el capítulo 3.2.1. Por último, se definió los valores de los grados de libertad que permitan simular cada uno de los 5 movimientos manuales.

Para importar el diseño de la mano ALARIS, se siguió los pasos ilustrados en la figura 3.58. El primer paso fue guardar el ensamblaje del archivo de SolidWorks como archivos STL teniendo en cuenta que el sistema de coordenadas de salida es el centro de la base de la mano ALARIS. El segundo paso fue importar los archivos STL en el software Blender para luego ser exportados como un archivo VRML2 (*Virtual Reality Modeling Language*). El último paso fue importar el archivo VRML2 en el simulador Webots® con la opción "*Import 3D Model*" ubicado en la pestaña "*File*".

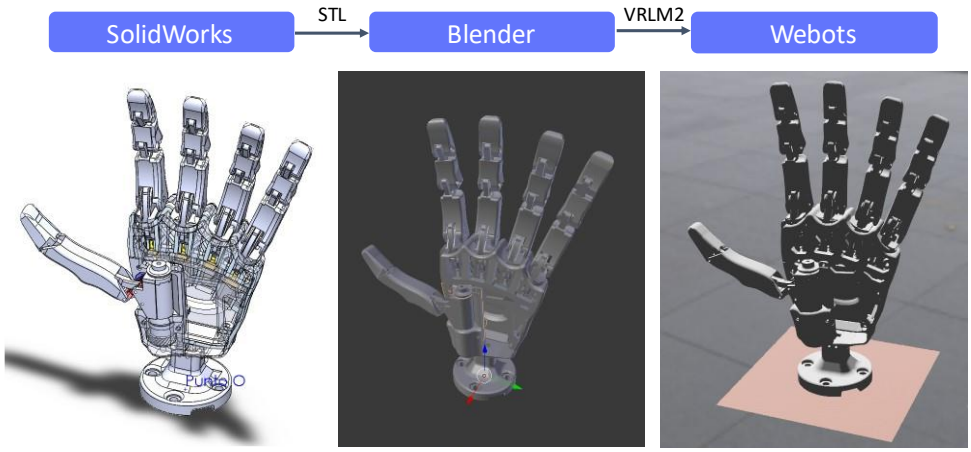


Figura 3.58. Pasos para importación del diseño a Webots®.
Fuente: Elaboración propia

Posterior a la importación, se configura los árboles nodales en Webots® para el movimiento de la mano ALARIS teniendo como base el nodo tipo *robot*. Todas las partes del diseño se convirtieron en nodos tipo *solid* (sólido) para luego ser agregarlos como *children* (hijo) al nodo tipo *robot*. En esta etapa se cambió la apariencia y color de los *solids* para una mejor distención entre las diferentes partes del diseño, tal como se muestra en la figura 3.59.

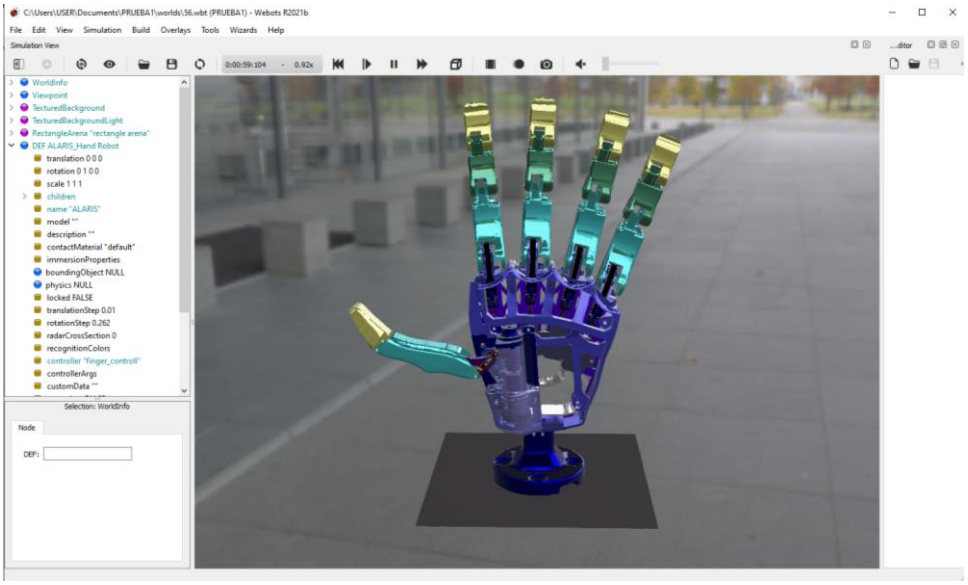


Figura 3.59. Configuración de los árboles nodales en Webots®.
Fuente: Elaboración propia

Para simular los movimientos rotacionales de los dedos mediales y el dedo pulgar se utilizaron nodos tipo *HingeJoint* (Junta de Bisagra) en las articulaciones que formaban los *solids*; por otro lado, para simular los movimientos deslizantes se utilizaron el nodo tipo *SliderJoint* (Junta Deslizante) en las cremalleras. Seguidamente, en los nodos tipo *HingeJoint* se agregaron los *devices* (dispositivos) *RotationalMotor* (Motor Rotacional) y *PositionSensor* (Sensor de posición); mientras que en el nodo tipo *SliderJoint*, se agregaron los *devices LinearMotor* (Motor Lineal) y *PositionSensor*. Cada *device* es representando por una variable en el programa del *controller* (controlador) del nodo *robot* con los fines de establecer las velocidades de los motores y obtener los valores de los sensores de posición.

A continuación, se programó con Python el control de posición de los motores en el *controller* siguiendo los diagramas de lazo de control de cada dedo medial y pulgar, mostrados en las figuras 3.60 y

3.61 respectivamente (ver ANEXO A.4). Las funciones de transmisión mecánica fueron calculadas en el capítulo 3.6.1 los cuales se muestran en las ecuaciones 3.12, 3.16, 3.20, 3.23, 3.28, 3.29, 3.32, 3.34 y 3.37.

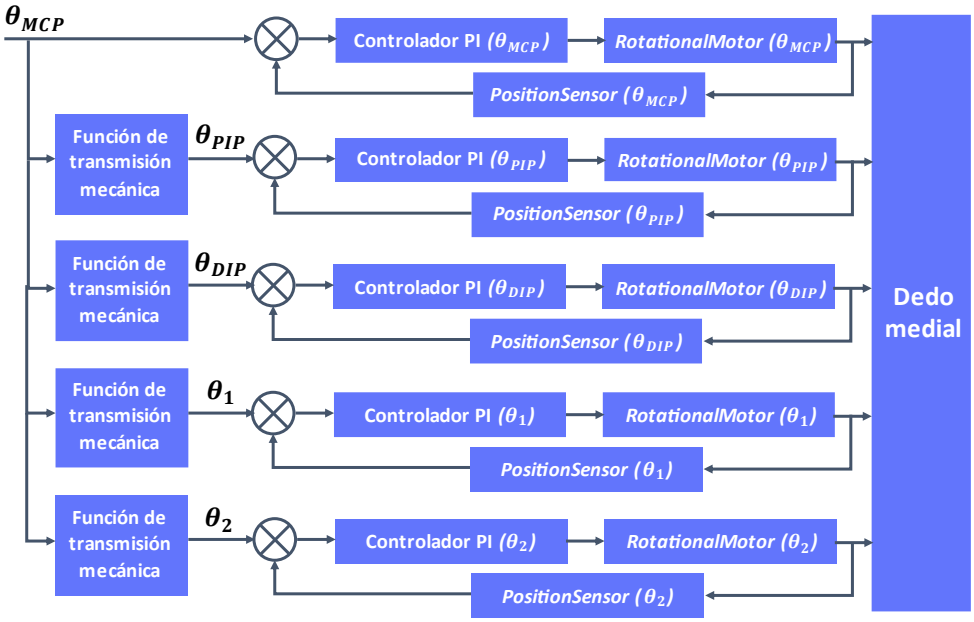


Figura 3.60. Diagrama de lazo para un dedo medial.
Fuente: Elaboración propia

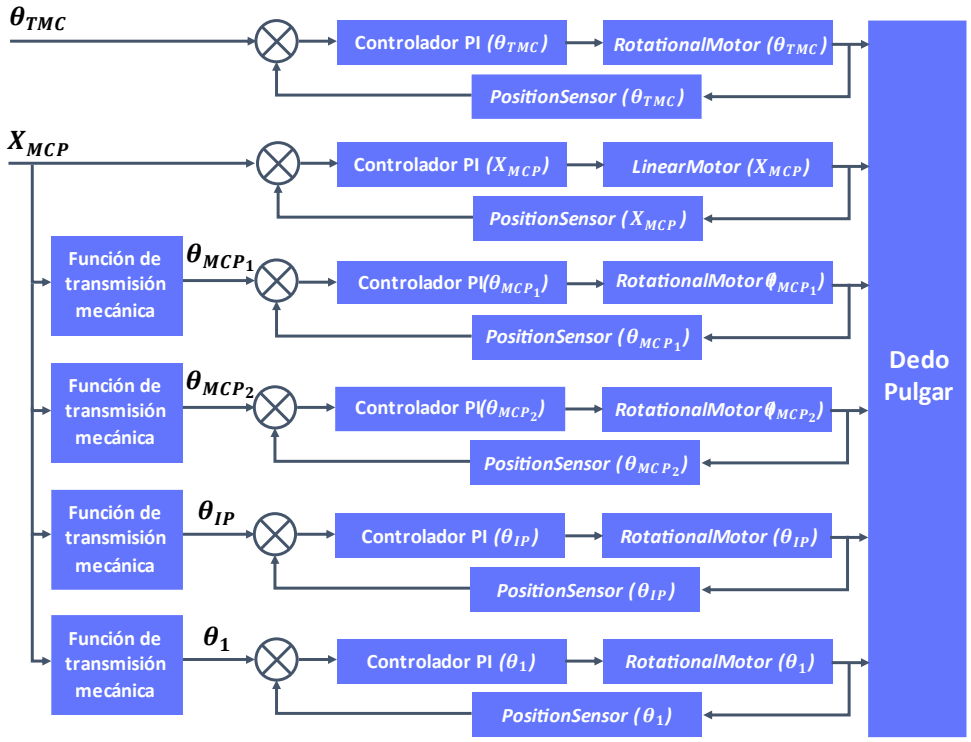


Figura 3.61. Diagrama de lazo para el dedo pulgar.
Fuente: Elaboración propia

Mediante diferentes pruebas de simulación, se seleccionó el controlador PI con ganancia proporcional igual a 4 y con ganancia integral igual 10^{-12} para cada uno de los motores con la finalidad de obtener respuestas sobreamortiguadas con tiempo de asentamiento aproximado de 0.65 segundos como se observa en la herramienta *Generic robot window* de Webots® (ver figura 3.62).

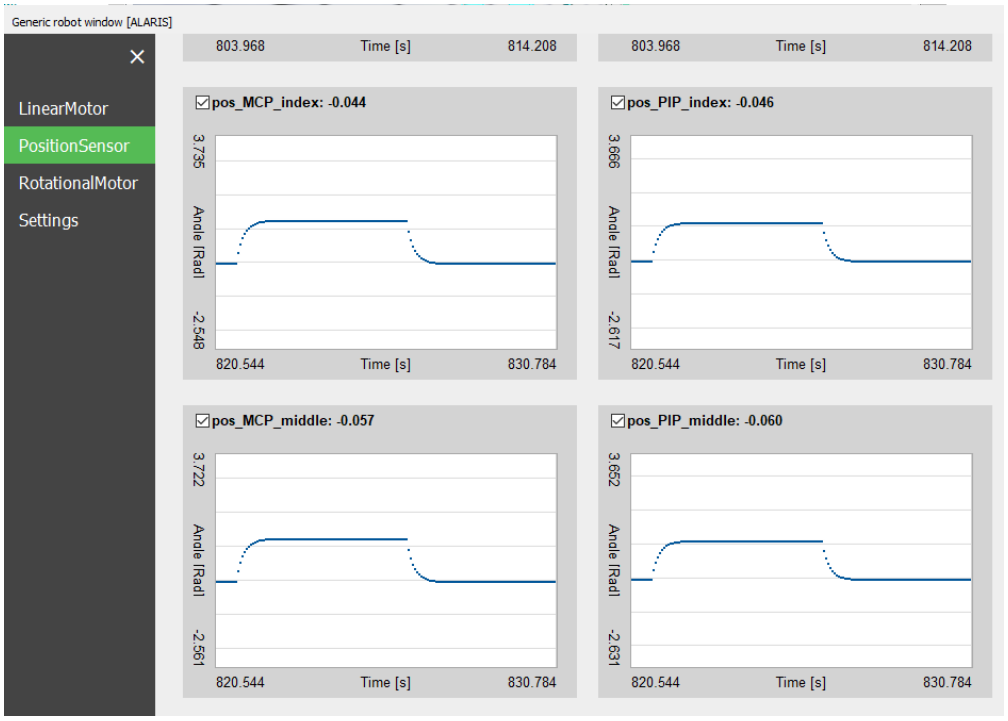


Figura 3.62. *Generic robot window* de la simulación de la mano ALARIS.
Fuente: Elaboración propia

Finalmente, se definió los valores de los ángulos θ_{MCP} de los dedos mediales y los valores del ángulo θ_{TMC} y la distancia X_{MCP} del dedo pulgar para simular cada movimiento manual, tal como se muestra en la tabla 3.15. Los resultados del ensamblaje de la mano ALARIS simulando los 5 movimientos manuales definidos anteriormente se observan en la figura 3.63.

Tabla 3.15. Valores de los Grados de libertad de la mano ALARIS.

Movimiento Manual	θ_{MCP} del dedo índice	θ_{MCP} del dedo medio	θ_{MCP} del dedo anular	θ_{MCP} del dedo meñique	θ_{TMC} del dedo pulgar	x_{MCP} del dedo pulgar
Mano relajada	10°	10°	10°	10°	15°	2°
Empuje de plataforma	-1°	-1	-1°	-1°	0°	0°
Pulgar aducido	65°	65°	65°	65°	0°	0°
Pinza lateral	80.65°	80.65°	80.65°	80.65°	0°	6.5°
Pinza pulgar-índice	40°	15°	15°	15°	84.4°	4.3°

Fuente: Elaboración propia

Figura 3.63. Simulación de los movimientos manuales.
Fuente: Elaboración propia

3.6.3. Comunicación del simulador con la GUI

La tercera etapa de la metodología del trabajo de (Forero et al., 2022) tiene como objetivo la adquisición y procesamiento de señales fisiológicas (EOG) para identificar movimientos oculares definidos. Mientras que, la cuarta etapa es la simulación de una prótesis en

Webots® conectado a Matlab con un control basado en la identificación de movimientos oculares para la realizar terapia de agarre de objetos.

En la presente tesis, se realiza una adaptación de la tercera y cuarta etapa de la metodología de (Forero et al., 2022) para cumplir con el objetivo de controlar un prótesis simulada en un entorno VR mediante la adquisición, procesamiento y clasificación de señales EMG para identificar los 5 movimientos manuales propuestos.

En el capítulo 3.5 se desarrolló una GUI para el control en tiempo real para una prótesis; asimismo en el subcapítulo 3.6.2 se programó la simulación de la prótesis realizando los diferentes movimientos manuales. Debido a que en ambos casos se programó en Python, la comunicación entre la simulación con la GUI se realizó mediante sockets de Python en el puerto 1233 de una misma red local. Para evitar interrupciones con otros procesos se decidió que el GUI y el simulador sean clientes de un servidor multihilo. En el ANEXO A.5 se muestra el código del servidor.

Por un lado, el servidor recibe y almacena el resultado de la clasificación en tiempo real de los movimientos manuales desde la GUI (cliente 1); y por el otro lado, el servidor envía el resultado almacenado en ese instante al simulador (cliente 2) en el mismo periodo de simulación. En la figura 3.63 se observa el flujo de los 3 sockets dónde el camino del resultado de la clasificación desde la GUI hasta el simulador se resalta en la línea anaranjada.

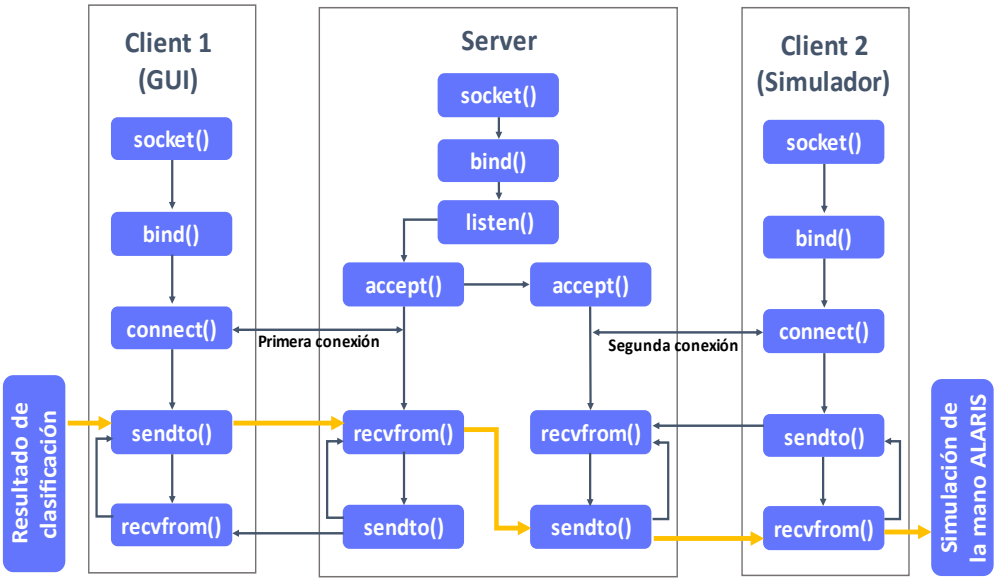


Figura 3.64. Flujograma de comunicación entre los 3 sockets.
Fuente: Elaboración propia

Finalmente, se realiza la simulación en Webots® del diseño la mano ALARIS controlado desde la GUI, el cual adquiere las señales EMG de un usuario realizando los movimientos manuales propuestos. En la figura 3.65 se observa la simulación del movimiento mano relajada; en la figura 3.66, empuje de plataforma; en la figura 3.67, pulgar aducido; en la figura 3.68, pinza lateral; y, en la figura 3.69, el movimiento pulgar índice.

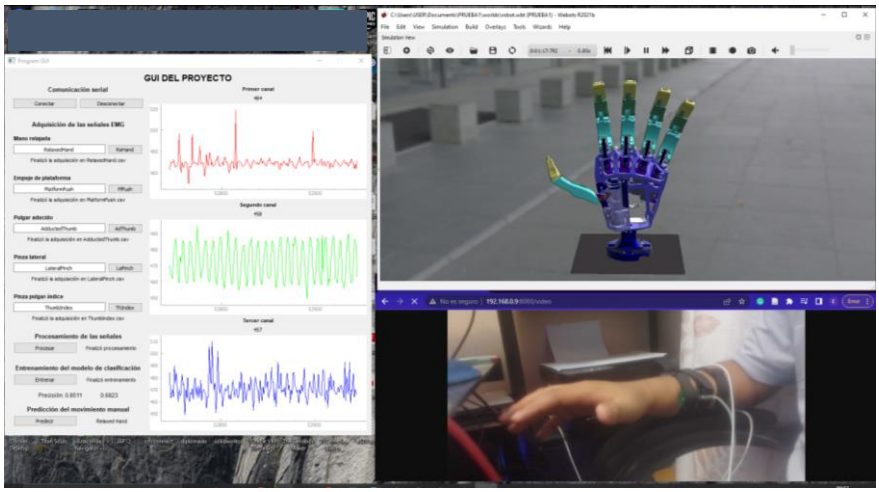


Figura 3.65. Simulación del movimiento mano relajada.
Fuente: Elaboración propia

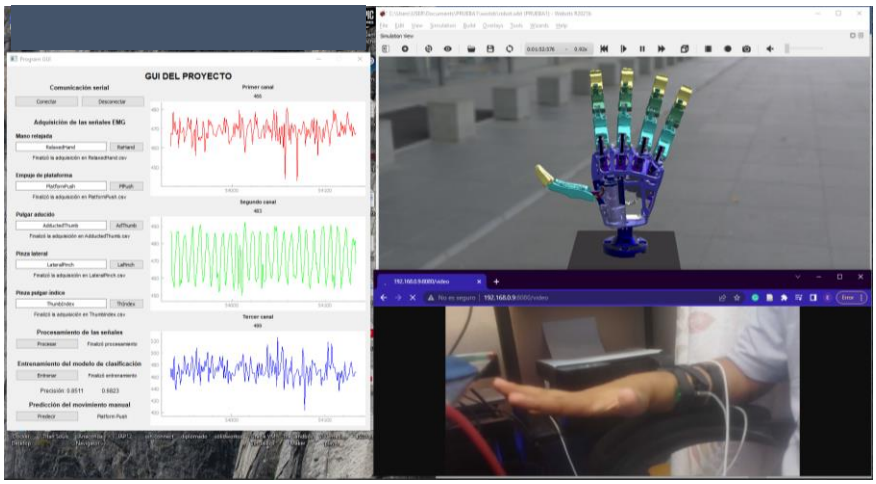


Figura 3.66. Simulación del movimiento empuje de plataforma.
Fuente: Elaboración propia

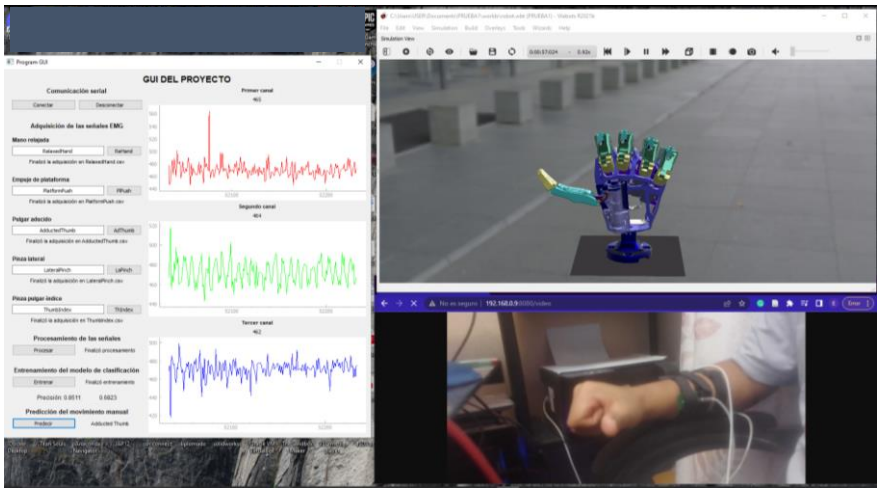


Figura 3.67. Simulación del movimiento pulgar aducido.
Fuente: Elaboración propia

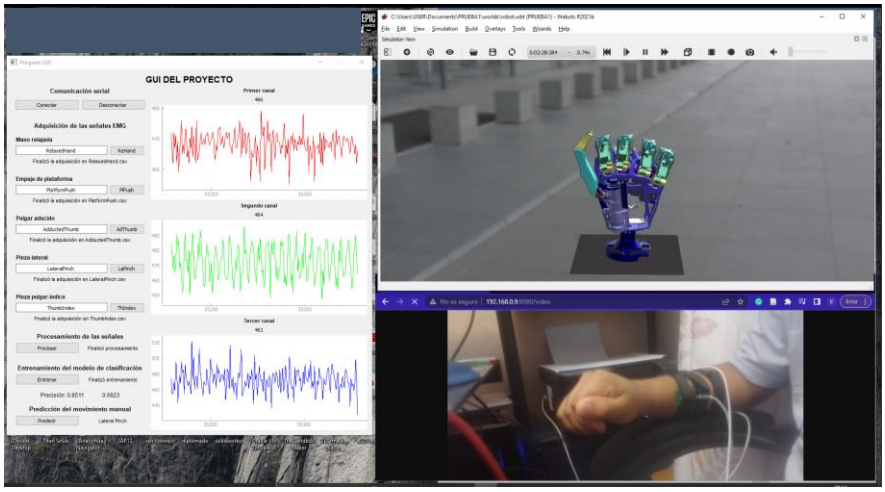


Figura 3.68. Simulación del movimiento pinza lateral.
Fuente: Elaboración propia

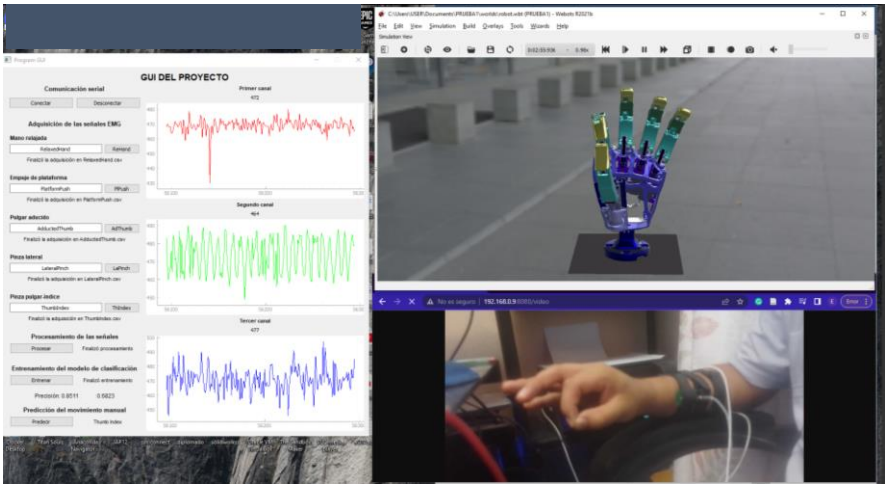


Figura 3.69. Simulación del movimiento pinza pulgar-índice.
Fuente: Elaboración propia

3.6.4. Conclusión de la simulación de una prótesis de mano

La simulación del diseño de código abierto de la mano ALARIS desarrollada a partir de la metodología de trabajo de (Forero et al., 2022) ha logrado cumplir los objetivos propuesto. Durante el análisis del diseño de los dedos mediales y del pulgar, se logró se comprender sus transmisiones de movimientos y simular sus cinemáticas mediante la convección D-H. En la etapa del ensamblaje, se logró importar el diseño desde el software SolidWorks® hasta en el software Webots® y programar los nodos para simular cada movimiento manual. Finalmente se logró la comunicación del simulador y la GUI de señales EMG mediante sockets de Python conectados a un servidor multihilo en una misma red local. Las figuras 3.65, 3.66, 3.67, 3.68 y 3.69 muestran que la simulación de los movimientos manuales de una prótesis de mano conectado a una GUI ha sido correctamente implementada generando una retroalimentación visual en el usuario.

CAPÍTULO IV ANÁLISIS Y DISCUSIÓN DE RESULTADOS

4.1. Resultados de las pruebas experimentales

Para validar el sistema de control mioeléctrico con clasificación en tiempo real de movimientos manuales basado en Deep Learning, se realizaron pruebas de destreza a voluntarios con la simulación del diseño de código abierto de la mano ALARIS conectada a la GUI. La realización de las pruebas contó con la participación de 5 voluntarios (4 hombres y 1 mujer) con un rango etario de 21 a 29 años, los cuales fueron informados previamente sobre la naturaleza de la investigación y su rol como participantes. Para ello se realizó la adquisición de las señales utilizando electrodos secos de los sensores de Wuxi Sichiray Co. posicionados en el antebrazo, tal como se definió en el diseño del subsistema de adquisición.

La configuración experimental para las pruebas de destreza se observa en la figura 4.1, el cual consiste en los siguientes elementos: (A) 3 placas de electrodos EMG, (B) placas de filtrados y amplificación, (C) tarjeta Raspberry Pi Pico para digitalizar las señales, (D) computadora portátil, (E) GUI de señales EMG, y (F) la simulación de la mano ALARIS.

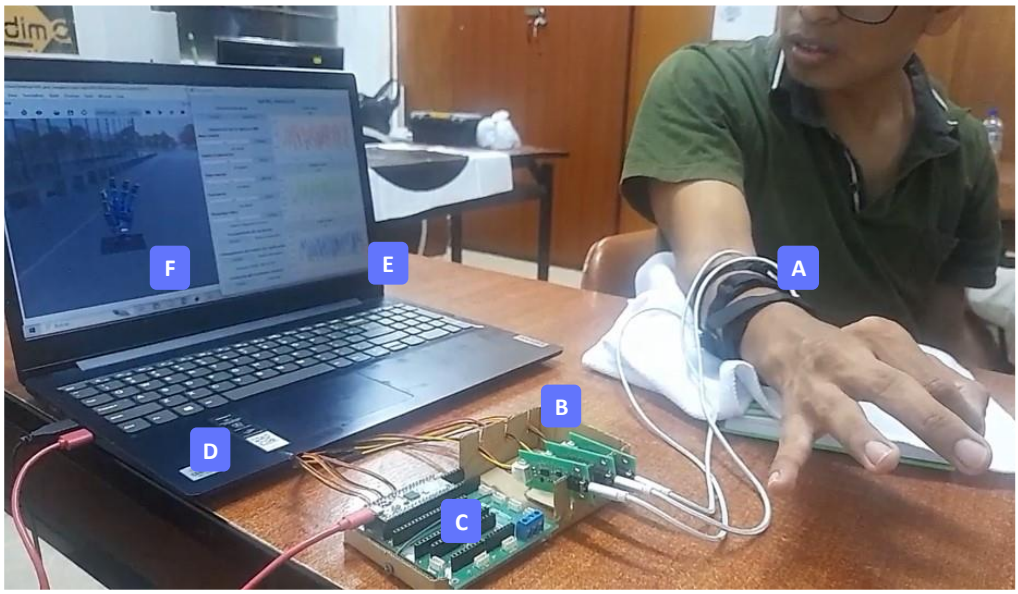


Figura 4.1. Configuración experimental para las pruebas destreza.
Fuente: Elaboración propia

Para realizar las pruebas de destreza, se siguieron tres pasos mostrados en la figura 4.2 para cada voluntario.

Figura 4.2. Flujograma del desarrollo del experimento.
Fuente: Elaboración propia

Durante el primer paso se prepara la configuración experimental, el cual abarca el posicionamiento de los electrodos en el antebrazo del voluntario y la ejecución de la GUI en la computadora portátil. El segundo paso es el proceso *Offline* el cual consiste en la adquisición de las señales mediante la GUI durante 60 segundos por cada movimiento manual, el procesamiento de las señales y el entrenamiento del modelo seleccionado durante el diseño del subsistema de clasificación (arquitectura *MobileNet* V2). El último paso es el proceso *Online* en el cual se realizan las pruebas de destreza.

La destreza para las prótesis es la capacidad que ayuda a los usuarios a realizar actividades de la vida diaria (AVD) (Gonzalez et al., 2015). En el desarrollo del marco conceptual de la tesis se planteó una estándar para comparar el grado de destreza que ofrece el sistema de control a las prótesis para capacitar la clasificación en tiempo real de los movimientos manuales de la taxonomía de sujeción de Cutkosky. Este estándar, que planteó en base a los trabajos de (Dollar, 2014) y (Cheu et al., 2005), se observa en la tabla 2.2 y la ecuación 2.8.

Por tal motivo, las pruebas de destreza, para la presente investigación, consisten en que cada voluntario realice los movimientos manuales en de forma secuencial según la programación mostrada en la figura 4.3, utilizando el sistema de control planteado. En las pruebas, los voluntarios deben mantener cada movimiento manual durante 20 segundos y relajar la mano antes de cada cambio de movimiento por 2 segundos.

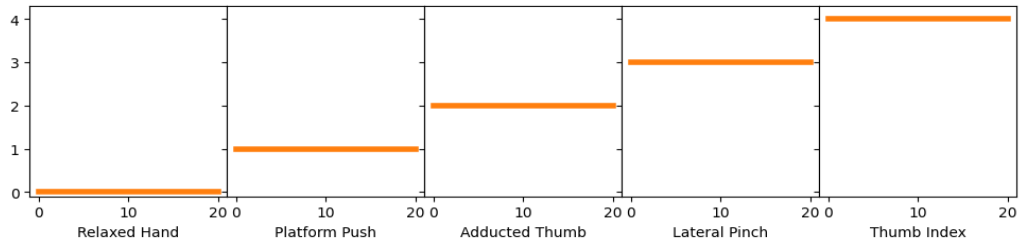


Figura 4.3. Programación de las pruebas de destreza.
Fuente: Elaboración propia

La finalidad de las pruebas, es adquirir la precisión de la clasificación en tiempo real por cada movimiento manual y en general; asimismo, la latencia de control de la simulación de la mano ALARIS con la GUI de señales EMG. Estos valores cuantificarán el rendimiento de las pruebas de destreza; y, según el resultado, validarán el sistema propuesto.

En el aula de investigación dónde se realizaron las pruebas experimentales se registró las condiciones ambientales de temperatura media de 23°C y una humedad media de 76%. A cada voluntario se le explicó detalladamente el procedimiento de las pruebas y se le proporciono un formulario de consentimiento de participación del experimento para la firma respectiva (ver ANEXO B).

4.1.1. Resultados del proceso *Offline*

El proceso *Offline* empieza con la adquisición de las señales EMG mediante la GUI, en este paso los voluntarios realizaron y mantuvieron cada movimiento manual durante 60 segundos. Los voluntarios pudieron descansar durante el tiempo que lo solicitaran entra cada adquisición. Las primeras ventanas de las señales adquiridas de cada voluntario según el movimiento manual se observan en la figura 4.4.

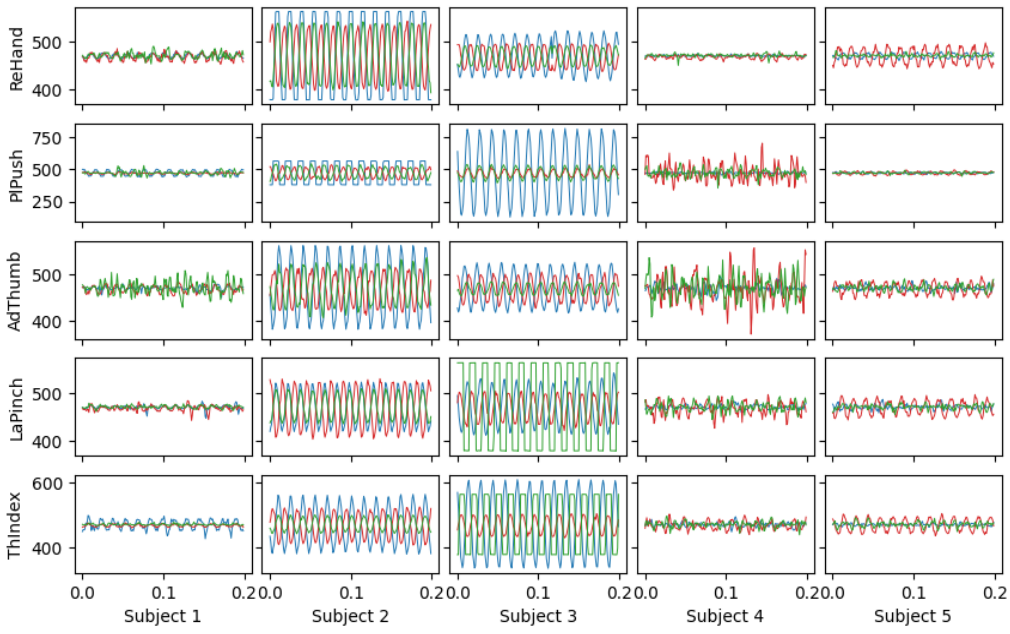


Figura 4.4. Ventanas de las señales EMG de los voluntarios.
Fuente: Elaboración propia

El segundo paso del proceso *Offline* abarca el procesamiento de las señales EMG mediante la GUI. En este paso las señales EMG de los voluntarios fueron divididas en ventanas de 200 ms, estandarizadas y filtradas, para luego ser transformadas en escalogramas, tal como se diseñó el subsistema de procesamiento. En la figura 4.5 se muestra los escalogramas del primer canal EMG de las primeras ventanas de las señales EMG de los voluntarios.

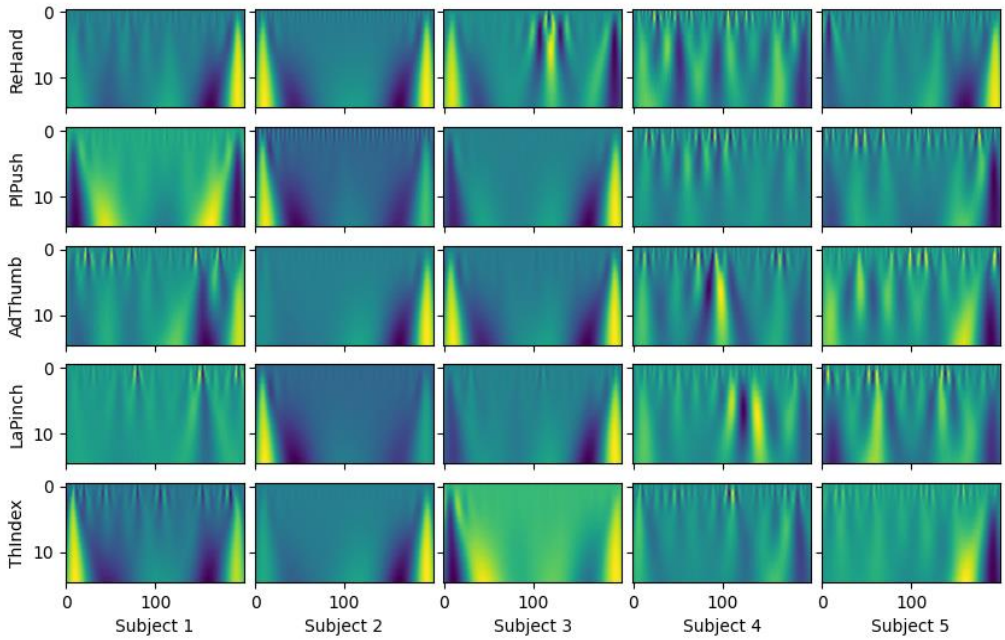


Figura 4.5. Escalogramas de las primeras ventanas de los voluntarios.
Fuente: Elaboración propia

El proceso *Offline* finaliza con el entrenamiento en una etapa del modelo de clasificación de arquitectura *MobileNet* V2 utilizando los pesos adquiridos mediante *Transfer Learning* durante el diseño del subsistema de clasificación.

Los resultados de los entrenamientos: la precisión y la pérdida de entropía cruzada (CEL) de la clasificación por cada voluntario y en general, se observan en la tabla 4.1.

Tabla 4.1. Resultados del proceso *Offline*.

Voluntario	**Precisión**	**CEL**
Subject 1	0.9889	0.0234
Subject 2	0.9733	0.0547
Subject 3	0.9822	0.056
Subject 4	0.9311	0.391
Subject 5	0.9156	0.0827
General	**0.9582**	**0.122**

Fuente: Elaboración propia

La precisión promedio de clasificación fue de 95.82% y la CEL promedio fue de 0.122 se muestran en la tabla 4.1. Ambos resultados nos permiten asegurar que la adquisición, procesamiento y clasificación de las señales EMG se realizaron de manera adecuada. Por añadidura, nos permite inferir que la clasificación en tiempo real también tendrá un resultado positivo.

4.1.2. Resultados de las pruebas de destreza

A partir del modelo entrenado y obtenido en el proceso *Offline*, la clasificación en tiempo real (proceso *Online*) es facultada para poder realizar las pruebas de destreza. La programación de las pruebas de destreza que los voluntarios ejecutaron se observa en la figura 4.3, la cual consiste en realizar y mantener cada movimiento manual durante 20 segundos.

Los resultados de las pruebas de destreza de cada voluntario se observan en la figura 4.6, donde el color naranja representa la clasificación esperada y el color azul representa la estimación en tiempo real.

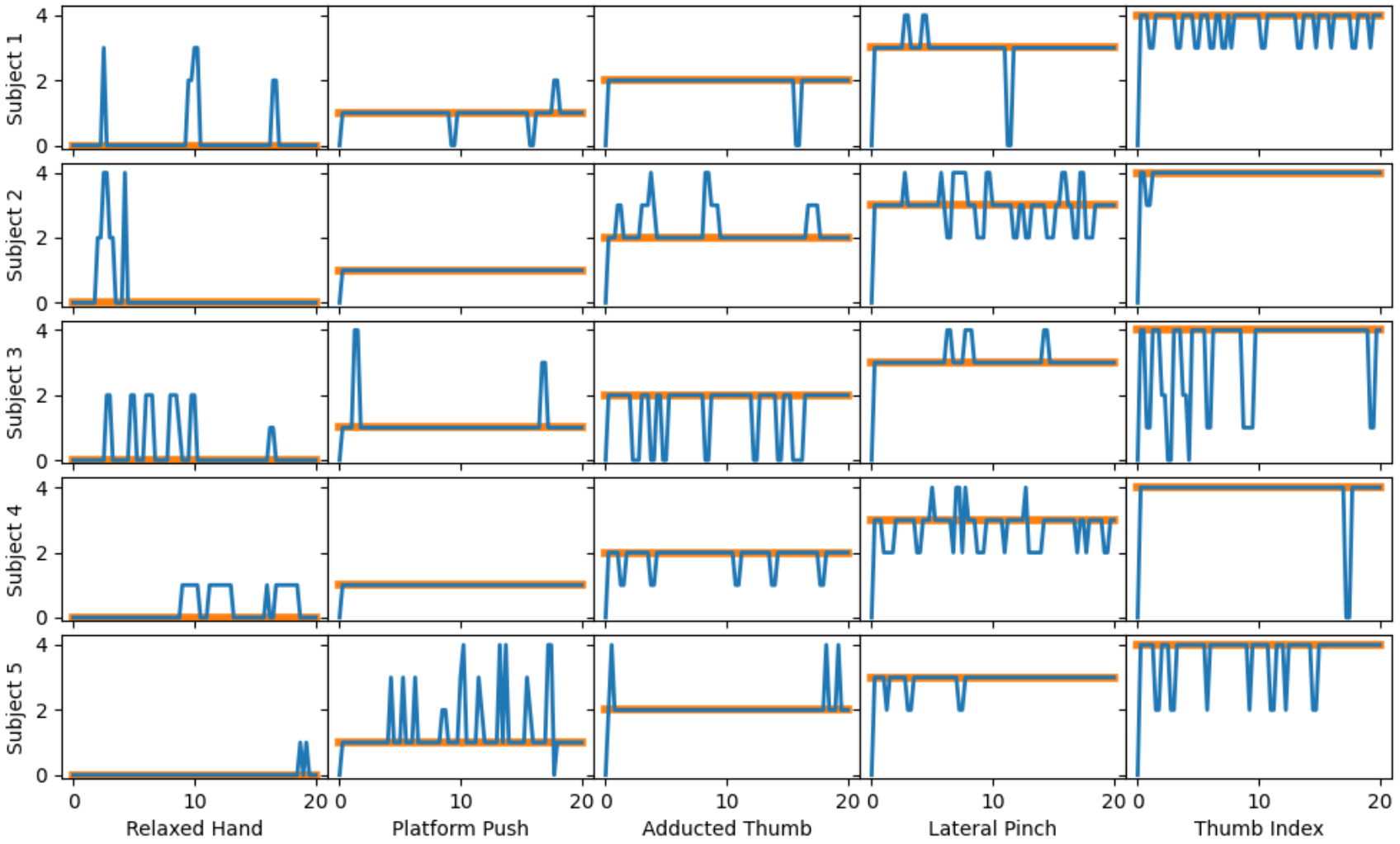


Figura 4.6. Resultados de las pruebas de destreza de cada voluntario.

Fuente: Elaboración propia

Los resultados cuantitativos de las pruebas de destreza se presentan mediante matrices de confusión en la tabla 4.2, dónde se observa la precisión por voluntario y por movimiento manual.

Tabla 4.2. Matrices de confusión de la clasificación en tiempo real.

Voluntario		ReHand	PlPush	AdThumb	LaPinch	ThIndex	Precisión
Subject 1	ReHand	**73**	0	4	3	0	**0.9125**
	PlPush	4	**74**	2	0	0	**0.925**
	AdThumb	2	0	**78**	0	0	**0.975**
	LaPinch	2	0	0	**74**	4	**0.925**
	ThIndex	0	0	0	20	**60**	**0.75**
Subject 2	ReHand	**73**	0	4	0	3	**0.9125**
	PlPush	0	**80**	0	0	0	**1**
	AdThumb	0	0	**64**	13	3	**0.8**
	LaPinch	0	0	17	**50**	13	**0.625**
	ThIndex	0	0	0	2	**78**	**0.975**
Subject 3	ReHand	**65**	3	12	0	0	**0.8125**
	PlPush	0	**76**	0	2	2	**0.95**
	AdThumb	17	0	**63**	0	0	**0.7875**
	LaPinch	0	0	0	**73**	7	**0.9125**
	ThIndex	3	10	4	0	**63**	**0.7875**
Subject 4	ReHand	**57**	23	0	0	0	**0.7125**
	PlPush	0	**80**	0	0	0	**1**
	AdThumb	0	10	**70**	0	0	**0.875**
	LaPinch	0	0	21	**54**	5	**0.675**
	ThIndex	2	0	0	0	**78**	**0.975**
Subject 5	ReHand	**78**	2	0	0	0	**0.975**
	PlPush	1	**64**	4	6	5	**0.8**
	AdThumb	0	0	**77**	0	3	**0.9625**
	LaPinch	0	0	5	**75**	0	**0.9375**
	ThIndex	0	0	11	0	**69**	**0.8625**

Fuente: Elaboración propia

Por otro lado, las medias de las precisiones por cada voluntario y en general de la clasificación en tiempo real para las pruebas de destreza se observan en la tabla 4.3.

Tabla 4.3. Precisiones de la clasificación en tiempo real.

Voluntario	ReHand	PlPush	AdThumb	LaPinch	ThIndex	**Promedio**
Subject 1	0.9125	0.925	0.975	0.925	0.75	**0.8975**
Subject 2	0.9125	1	0.8	0.625	0.975	**0.8625**
Subject 3	0.8125	0.95	0.7875	0.9125	0.7875	**0.85**
Subject 4	0.7125	1	0.875	0.675	0.975	**0.8475**
Subject 5	0.975	0.8	0.9625	0.9375	0.8625	**0.9075**
General	**0.865**	**0.935**	**0.88**	**0.815**	**0.87**	**0.873**

Fuente: Elaboración propia

El movimiento manual más difícil de estimar es la Pinza Lateral (*Lateral Pinch*) debido a la precisión de 81.5% en promedio y la precisión para el voluntario 4 de 62.5%, tal como se observa en la tabla 4.3. Por otro parte, el movimiento manual Empuje de Plataforma (*Plataform Push*) fue el más estimable con un valor promedio de precisión de 93.5% y 100% para los voluntarios 2 y 4. La precisión individual más alta fue 90.75% y la más baja fue 84.75%, correspondientes a los voluntarios 5 y 4, respectivamente. El promedio de las precisiones de los 5 voluntarios fue del 87.3%.

Con respecto a la latencia durante la clasificación de tiempo real, se calculó en función a la diferencia de tiempo entre estimaciones de los movimientos manuales. La latencia promedio de cada voluntario se observa en la figura 4.7, en cual el voluntario 1 tuvo menor latencia con un valor de 260 ms. La latencia promedio de los voluntarios corresponde al valor de 276.97 ms.

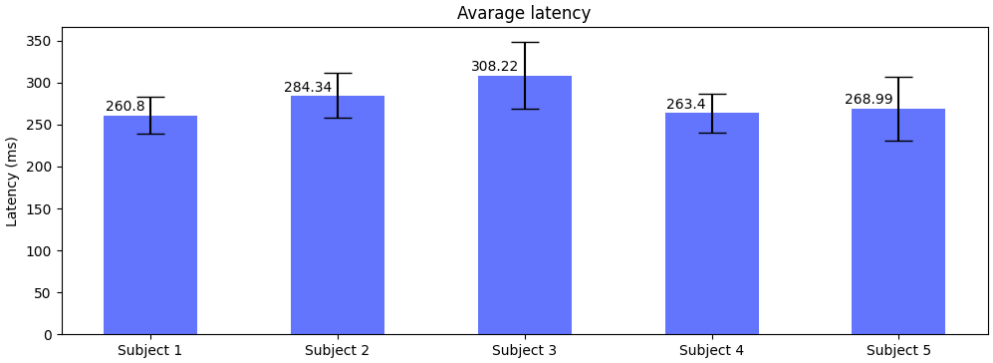


Figura 4.7. Latencia promedio por voluntario.
Fuente: Elaboración propia

4.1.3. Destreza que ofrecen los sistemas de control

Las prótesis de mano de agarre simple tienen un sistema de control que proporciona la funcionalidad para abrir y cerrar los dedos la mano (Chappell et al., 1987). Ello permite al usuario sostener, levantar, manipular objetos de limitadas formas para ejecutar distintas actividades de la vida diaria (AVD). Este tipo de control normalmente se divide en control de encendido/apagado y control proporcional de intensidad de las señales EMG (Geethanjali, 2016).

El control encendido/apagado permite a una prótesis realizar el movimiento empuje de plataforma durante la abertura de los dedos (C_1= 0.1). Durante la cerradura de los dedos, este tipo de control permite realizar el movimiento diámetro largo (agarre cilíndrico) o pulgar aducido (C_2= 0.14).

A continuación, se calcula en la tabla 4.4 y ecuación 4.1 el grado de destreza que ofrecen el sistema de control encendido/apagado según el estándar elaborado en la tabla 2.2 y la ecuación 2.8, asumiendo que su precisión de clasificación en tiempo real de ambos movimientos manuales es de 100%.

Tabla 4.4. Valores del estándar para el control de encendido/apagado.

C_i (coeficiente)	Valor	P_i (precisión)	Valor
C_1	0.1	P_1	1
C_2	0.15	P_2	1
C_3	0	P_3	0
C_4	0	P_4	0
C_5	0	P_5	0

Fuente: Elaboración propia

$$\Delta Destreza_{on/off} = 0.25 \quad (4.1)$$

El control proporcional a la intensidad de las señales EMG permite a una prótesis realizar los mismos movimientos que el control de encendido/apagado (C_1= 0.1 y C_2= 0.15). Adicionalmente, debido a su control proporcional, puede clasificar un estado intermedio que sostenga objetos con 3 dedos sin apoyarse en la palma (C_4= 0.25).

En la tabla 4.5 y ecuación 4.2 se calcula el grado de destreza que ofrecen el sistema de control proporcional según el estándar elaborado en la tabla 2.2 y la ecuación 2.8, asumiendo que el control logra reflejar completamente la intensidad de las señales EMG.

Tabla 4.5. Valores del estándar para el control proporcional.

C_i (coeficiente)	Valor	P_i (precisión)	Valor
C_1	0.1	P_1	1
C_2	0.15	P_2	1
C_3	0	P_3	0
C_4	0.25	P_4	1
C_5	0	P_5	0

Fuente: Elaboración propia

$$\Delta Destreza_{prop} = 0.50 \quad (4.2)$$

El sistema de control con clasificación de movimientos manuales propuesto en la presente de investigación ha logrado predecir 4 movimientos que se incluyen en la taxonomía según Cutkosky para 5 voluntarios con una precisión mostrada en la tabla 4.3.

A continuación, se calcula en la tabla 4.6 y ecuación 4.3 el grado de destreza que ofrecen el sistema de control propuesto según el estándar elaborado en la tabla 2.2 y la ecuación 2.8.

Tabla 4.6. Valores del estándar para el control propuesto.

C_i (coeficiente)	Valor	P_i (precisión)	Valor
C_1	0.1	P_1	0.935
C_2	0.15	P_2	0.88
C_3	0.2	P_3	0.815
C_4	0	P_4	0
C_5	0.3	P_5	0.87

Fuente: Elaboración propia

$$\Delta Destreza_{mov} = 0.6495 \quad (4.3)$$

4.2. Contrastación de hipótesis

La hipótesis general menciona que el diseño de un sistema de control mioeléctrico con clasificación en tiempo real de movimientos manuales basado en Deep Learning mejorará la destreza de las prótesis de mano de agarre simple. Se planteó un estándar para comparar el grado de destreza que ofrece el sistema de control a las prótesis de mano en base a los trabajos de los trabajos de (Cheu et al., 2005) y (Dollar, 2014). El estándar propone comparar los sistemas de control que clasifiquen los movimientos manuales de la taxonomía de sujeción de Cutkosky, los cuales están relacionados

directamente a las actividades de la vida diaria (AVD). Este enfoque está alineado a la definición de la destreza como una capacidad de la mano para realizar AVD (Gonzalez et al., 2015). En consecuencia, se propuso un sistema que adquiera, procese y clasifique en tiempo real señales EMG de 4 movimientos manuales de la taxonomía de Cutkosky y el movimiento de la mano relajada, con el objetivo de mejorar la destreza ofrecida por los sistemas de control de agarre simple. Mediante pruebas de destreza a 5 voluntarios, el sistema planteado obtuvo una precisión media de clasificación en tiempo real de 87.3%, asimismo, el valor de destreza obtenido según el estándar propuesto (ver ecuación 4.3) fue de 0.6495. Este valor es mayor el grado de destreza que ofrecen los sistemas de control encendido/apagado (0.25) y proporcional a las señales EMG (0.5) pertenecientes a las prótesis de agarre simple en un 160% y 30% respectivamente (ver ecuación 4.1 y 4.2). La implementación del diseño del sistema de control propuesto en una prótesis capacitaría al usuario a realizar una mayor variedad de AVD como el agarre y manipulación de tarjetas, papeles, billetes, monedas, etc., mejorando su destreza respecto a una prótesis de agarre simple. Por lo tanto, se valida en forma positiva la hipótesis general del trabajo de investigación.

En la primera hipótesis específica se menciona que la selección de criterios y técnicas de adquisición y procesamiento de señales EMG ayudará a obtener escalogramas favorables en el entrenamiento de clasificadores de movimientos manuales. El subsistema de adquisición logró tener un correcto diseño debido a los criterios de selección del sensor EMG y número de canales y las técnicas de posicionamiento de electrodos y digitalización,

permitiendo obtener señales de los movimientos manuales de los 5 voluntarios distinguibles entre sí (ver figura 4.4). Por otro lado, el subsistema de procesamiento logró transformar correctamente las señales EMG de los voluntarios en escalogramas mediante las técnicas de *Windowing*, estandarización, filtrado y la CWT (ver figura 4.5). En la tabla 4.1 se muestra que el subsistema de adquisición y procesamiento permitió obtener una precisión de clasificación después de los entrenamientos de los escalogramas para el voluntario 1 y en general de 98.89% y 95.82% respectivamente, validando así la primera hipótesis específica.

En la segunda hipótesis específica se menciona que la comparación de clasificadores a partir de escalogramas estimará en tiempo real la intención de movimiento manual. Durante el diseño del subsistema de clasificación se comparó 80 modelos utilizando 4 arquitecturas de Deep Learning y al variar los parámetros de la CWT; partiendo de ello, se seleccionó el modelo con la mayor precisión y menor pérdida de entropía cruzada (CEL). Este modelo se utilizó en el proceso *Offline* de las pruebas experimentales con los 5 voluntarios, y se obtuvo una precisión media de 95.82% y una CEL promedia de 0.122 después de los entrenamientos como se observa en la tabla 4.1; el bajo valor de CEL indica que no se producido un sobreajuste en el entrenamiento de los modelos. Por otro lado, en la tabla 4.3 se muestra como a partir de las pruebas de destreza con el modelo seleccionado se estimó en tiempo real la intención en movimiento manual con una precisión de 90.75% y 87.3% para el voluntario 5 y en general, respectivamente. A partir de estos resultados experimentales, se valida la segunda hipótesis específica.

En la tercera hipótesis específica se menciona que la validación del sistema de control será posible a través de pruebas de destreza con la simulación de una prótesis conectado a una GUI de señales EMG. En el capítulo 3.5, se diseñó una GUI que permite al usuario adquirir y procesar señales EMG para el entrenamiento del modelo de clasificación seleccionado; asimismo, después del entrenamiento, permitir la estimación en tiempo real de los movimientos manuales. Por otro lado, en el capítulo 3.6, se simuló el diseño de código abierto de la mano ALARIS en el entorno virtual de Webots®, el cual se conectó mediante sockets de Python con servidor multihilo a la GUI para simular los movimientos manuales estimados y de esta manera generar una retroalimentación visual en el usuario. El desarrollo de las pruebas de destreza a 5 voluntarios mediante la simulación y la GUI tuvieron una precisión y latencia de control media de 87.3% (ver tabla 4.3) y 276 ms (ver figura 4.7). Estos resultados validaron el sistema de control debido a que mejoró la destreza de las prótesis de agarre simple (ver ecuaciones 4.1, 4.2 y 4.3) y su valor de latencia de control es menor a los 300 ms, tal como recomienda el estado del arte (Nahid et al., 2020a). Por lo tanto, la tercera hipótesis específica se valida positivamente.

CONCLUSIONES

En consideración de los resultados experimentales, se afirma que el sistema de control mioeléctrico con clasificación en tiempo real de movimientos manuales basado en Deep Learning, desarrollado en el presente trabajo de investigación, cumple con los objetivos generales y específicos propuestos, concluyendo que:

1. El diseño de un sistema de control mioeléctrico con clasificación en tiempo real de movimiento manuales basado en Deep Learning ha logrado mejorar la destreza de una prótesis de mano de agarre simple a través de técnicas de adquisición, procesamiento y clasificación de señales EMG. Adicionalmente, la programación de una Interfaz Gráfica de Usuario (GUI), para que el usuario interactúe eficientemente con estás técnicas, y la simulación de una prótesis virtual, para lograr una retroalimentación visual, han permitido la ejecución de pruebas de destreza a 5 voluntarios para evaluar la clasificación en tiempo real de 5 movimientos manuales vinculadas estrechamente a las actividades de la vida diaria (AVD). Estas pruebas tienen como base teórica a un estándar de destreza que cuantifica el grado de destreza ofrecido por un sistema de control hacia una prótesis de mano, y consisten en que un usuario realice de forma secuencial cada movimiento durante 20 segundos con la finalidad de obtener la precisión de estimación cuando se emplea el sistema de control propuesto. Los resultados fueron alentadores ya que se logró una precisión media de

clasificación en tiempo real de 87.3%, significando una mejora de al menos 30% del grado de destreza que ofrecen los sistemas de control de agarre simple de las prótesis de mano según el estándar de destreza. Esto se refleja en la capacidad que ofrece el sistema de control propuesto al usuario a realizar una mayor variedad de AVD como la manipulación de objetos laminares o pequeños, lo que supone fomentar la independencia y elevar el nivel de vida de los pacientes. En el mismo sentido, los resultados avalan al sistema propuesto como una alternativa ante la necesidad de soluciones asequibles y efectivas para el control de prótesis de alta destreza en el Perú. Por las razones anteriormente mencionadas, se afirma que se ha logrado el objetivo general de la tesis.

2. La selección meticulosa de técnicas de adquisición y procesamiento de señales electromiográficas (EMG) ha demostrado ser crucial para obtener escalogramas favorables en el entrenamiento de clasificadores de movimientos manuales. La selección del sensor EMG y número canales, asimismo, las técnicas de posicionamiento de electrodos y digitalización han permitido un adecuado diseño del subsistema de adquisición. En el mismo sentido, la técnica de *Windowing*¸ estandarización, filtrado de señales y la CWT han posibilitado que las señales sean transformadas a escalogramas distinguibles entre sí para cada movimiento manual. Los resultados experimentales respaldan una correcta selección de técnicas para la adquisición y procesamiento de señales a virtud de las precisiones de clasificación logradas después del entrenamiento de los escalogramas las cuales son de 98.89% para un voluntario y 95.82% en promedio.

3. La comparación de clasificadores a partir de escalogramas de señales EMG ha permitido estimar en tiempo real la intención de movimiento manual. Durante el proceso de diseño del subsistema de clasificación, se evaluaron 80 modelos a partir de 4 arquitecturas de Deep Learning variando los parámetros de la CWT con el fin de seleccionar un modelo con alta precisión de clasificación y baja pérdida de entropía cruzada. El modelo seleccionado (arquitectura *MobileNet,* el valor de escala de 15 y función madre *Mexican Hat*) aseguró un equilibrio entre la precisión y generalización aceptable para el control de una prótesis de mano. Esta conclusión es respaldada por los resultados de las pruebas experimentales, con una precisión de estimación en tiempo real de 5 movimientos manuales de 90.75% para un voluntario y 87.3% en general.
4. El sistema de control se validó correctamente a través de las pruebas de destreza con la simulación de la mano ALARIS conectada a la GUI de señales EMG. El diseño y la implementación de la GUI permitió adquirir, procesar y clasificar en tiempo real la señales EMG; asimismo, la simulación de la prótesis permitió que los voluntarios puedan observar inmediatamente el movimiento manual que estima el sistema propuesto resultando en una retroalimentación visual. La validación se sustenta en los resultados de las pruebas destreza realizadas a 5 voluntarios, dónde se obtuvo una precisión en tiempo real y una latencia de control media de 87.3% y 276 ms respectivamente. La alta precisión significó que el sistema de control propuesto mejoró la destreza de las prótesis de agarre simple; y el valor de la latencia, que el control se realiza en tiempo real.

RECOMENDACIONES

Las recomendaciones para próximos proyectos en relación al sistema de control de prótesis de mano planteado, se enuncian a continuación:

1. Una recomendación clave para la implementación del sistema de control propuesto es considerar su integración en una tarjeta de Raspberry Pi. Esta elección se basa en su alta potencia de procesamiento, famosa por su capacidad para ejecutar modelos de Deep Learning; asimismo, por la posibilidad de su programación en Python, lenguaje que se utilizó a largo de la tesis. La adopción de esta plataforma no solo promovería la portabilidad del sistema de control con la prótesis, sino que también estaría alineada con las tendencias actuales de desarrollo de dispositivos de alto rendimiento y bajo costo.
2. En consonancia al enfoque de lograr un sistema altamente portátil y de interacción intuitiva, se recomienda explorar la opción de reemplazar la GUI actual por alternativas más versátiles para las funciones claves como la administración durante la adquisición de las señales EMG para cada movimiento manual. La adopción de comandos de voz podría significar una interacción más natural y práctica, especialmente cuando el paciente se encuentre solo o mantiene la otra mano ocupada.
3. Una vía prometedora para potenciar la destreza brindada por el sistema de control, radica en la expansión del repertorio de movimientos manuales a clasificar. Para concretar esta idea, se podría considerar el incremento

en el número de sensores EMG ubicados en el antebrazo. Esto permitiría un enriquecimiento de la información de entrada para subsistema de clasificación. Asimismo, se plantea el uso de otro tipo de sensores como las unidades de medición inercial (IMU), lo que podría ofrecer una representación aún más completa de los movimientos manuales.

4. Para abordar las molestias expresadas por los voluntarios sobre la presión ejercida por las correas de los electrodos (ver ANEXO B), se recomienda diseñar brazaletes personalizados utilizando tecnología de impresión 3D. Estos brazaletes se ajustarían cómodamente alrededor de los músculos, proporcionando una presión adecuada para el correcto funcionamiento de los electrodos. Esta medida no solo mejoraría la comodidad del usuario, sino que también podría aumentar la precisión de medición, ya que los electrodos mantendrían una posición fija por un mayor tiempo.
5. Se siguiere considerar la implementación del sistema de control con una prótesis de mano física y posteriormente validar su funcionamiento mediante las pruebas de destreza en pacientes con amputación. Esta implementación permitiría obtener datos concretos sobre el desempeño del sistema en un entorno clínico y evaluar directamente su impacto en el nivel de vida de los usuarios a través de un seguimiento continuo para ajustar y optimizar aún más el sistema de control, adaptándolo a las preferencias y necesidades individuales.
6. Adicionalmente, se plantea la posibilidad de explorar una extrapolación de la metodología de diseño hacia interfaces cerebro-computadora (BCI). Si bien esta sugerencia abre un horizonte interesante, es necesario abordarla

con precaución. La integración de BCI en sistemas de prótesis podría revolucionar la interacción entre el usuario y la prótesis, ofreciendo un nivel de control más intuitivo y preciso. Sin embargo, es esencial tener en cuenta que las señales electromiográficas (EMG) y electroencefalográficas (EEG) presentan diferencias significativas en términos de adquisición y procesamiento. Las BCI pueden ser más complejas para lograr una estabilidad de las señales, y pueden requerir equipos y conocimientos especializados. Por ello, esta recomendación necesita ser evaluada minuciosamente en función de los recursos disponibles y las perspectivas futuras del campo del control de las prótesis.

REFERENCIAS BIBLIOGRÁFICAS

Aceves-Fernandez, M. A. (2019). *Artificial Intelligence—Applications in Medicine and Biology*.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Van Esesn, B. C., Awwal, A. A. S., & Asari, V. K. (2018). *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches* (arXiv:1803.01164). arXiv.

Alvarado Castillo, V., Sánchez Flores, J., Gómez, J. C., Chihuan Huayta, E., De La Cruz Casaño, C., Alvarado Castillo, V., Sánchez Flores, J., Gómez, J. C., Chihuan Huayta, E., & De La Cruz Casaño, C. (2019). Adquisición de señales SEMG con electrodos secos para el control de movimiento de dedos en una prótesis robótica fabricada en una impresora 3D. *Ingeniare. Revista chilena de ingeniería*, *27*(3), 522-536.

Alzate Arias, N. (2018). *Control mioeléctrico de una prótesis de miembro superior - Mano*.

Belkhou, A., Jbari, A., & Belarbi, L. (2017). A continuous wavelet based technique for the analysis of electromyography signals. *2017 International Conference on Electrical and Information Technologies (ICEIT)*, 1-5.

Bohórquez Bendezú, J. L. (2021). *Diseño del sistema de control de una prótesis transfemoral a partir del reconocimiento de la intención de movimiento*.

Briones Manrique, C. J. (2020). Sistema portátil de monitoreo de señales electromiográficas a través de electrodos de superficie. *Repositorio Académico USMP*.

Brito Guaricela, J. L., Quinde Abril, M. X., & Cuzco Patiño, J. D. (2013). *Diseño, construcción e implementación de una prótesis biomecánica de mano derecha*.

Bustamante Carvallo, M. M. (2018). *Malky: Diseño e implementación de una prótesis parcial de mano personalizada*.

Calado, A., Soares, F., & Matos, D. (2019). A Review on Commercially Available Anthropomorphic Myoelectric Prosthetic Hands, Pattern-Recognition-Based Microcontrollers and sEMG Sensors used for Prosthetic Control. *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 1-6.

Calderon-Cordova, C., Ramírez, C., Barros, V., Quezada-Sarmiento, P. A., & Barba-Guamán, L. (2016). EMG signal patterns recognition based on feedforward Artificial Neural Network applied to robotic prosthesis myoelectric control. *2016 Future Technologies Conference (FTC)*, 868-875.

Camacho Conchucos, H. T. (2010). Pacientes amputados por accidentes de trabajo: Características y años acumulados de vida productiva potencial perdidos. *Anales de la Facultad de Medicina*, *71*(4), 271-275.

Chappell, P. H., Nightingale, J. M., Kyberd, P. J., & Barkhordar, M. (1987). Control of a single degree of freedom artificial hand. *Journal of Biomedical Engineering*, *9*(3), 273-277.

Cheu, L. E. R., Casals, A., Cuxart, A., & Parra, A. (2005). Towards the definition of a functionality index for the quantitative evaluation of hand-prosthesis. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 541-546.

CWT—PyWavelets Documentation. (s. f.). Recuperado 29 de diciembre de 2022, de https://pywavelets.readthedocs.io/en/latest/ref/cwt.html

Day, S. (2002). Important factors in surface EMG measurement. *Bortec Biomedical Ltd Publishers*, 17.

Denavit, J., & Hartenberg, R. S. (1955). A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *Journal of Applied Mechanics*, *22*(2), 215-221.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248-255.

Deng, L., & Yu, D. (2014). Deep Learning: Methods and Applications. *Foundations and Trends® in Signal Processing*, *7*(3–4), 197-387.

Dollar, A. M. (2014). Classifying Human Hand Use and the Activities of Daily Living. En R. Balasubramanian & V. J. Santos (Eds.), *The Human Hand as an Inspiration for Robot Hand Development* (pp. 201-216). Springer International Publishing.

Dorf, R. C., Poularikas, A. D., & Grigoryan, A. M. (2018). *Transforms and Applications Handbook* (3.ª ed.). CRC Press.

Drake, R. L., Mitchell, A. M. W., & Vogl, A. W. (2020). *Gray. Anatomía para estudiantes*. Elsevier Health Sciences.

Earle F., P. I. (2022). *Arduino-Pico—Arduino-Pico 2.7.0 documentation*. https://arduino-pico.readthedocs.io/en/latest/

Englehart, K., & Hudgins, B. (2003). A robust, real-time control scheme for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, *50*(7), 848-854.

Fazeli, M., Karimi, F., Ramezanian, V., Jahanshahi, A., & Seyedin, S. (2020). Hand Motion Classification Using sEMG Signals Recorded from Dry and Wet Electrodes with Machine Learning. *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, 1-4.

Flanagan, R., & Johansson, R. S. (2002). Hand Movements. *Encyclopedia of the Human Brain*, *12*, 1-16.

Forero, B., Velásquez, K., Hernández, R., & Mejía, E. (2022). Simulation of transradial prosthesis using Virtual Reality Environment and electrooculography (EOG) signals for grip therapy. *Visión Electrónica*, *16*(2), Article 2.

Fu, Y., Zhao, J., Dong, Y., & Wang, X. (2020). Dry Electrodes for Human Bioelectrical Signal Monitoring. *Sensors*, *20*(13), Article 13.

Galarza Flores, M. C. (2018). Clasificación de señales mioeléctricas superficiales de los movimientos de la mano utilizando técnicas de aprendizaje supervisado. *Universidad Nacional de San Agustín de Arequipa*.

Gallardo, S. (2019). *Elementos de sistemas de telecomunicaciones 2.ª edición 2019*. Editorial Paraninfo.

Garcia Mora, M. E., Schwartz Orellana, S., & Freire, G. (2021). *Disability Inclusion in Latin America and the Caribbean: A Path To Sustainable Development*. World Bank.

Geethanjali, P. (2016). Myoelectric control of prosthetic hands: State-of-the-art review. *Medical Devices (Auckland, N.Z.)*, *9*, 247-255.

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). Fundamental Concepts of Convolutional Neural Network. En V. E. Balas, R. Kumar, & R. Srivastava (Eds.), *Recent Trends and Advances in Artificial Intelligence and Internet of Things* (pp. 519-567). Springer International Publishing.

González, J. M. V. (2020). *Sistematización de prótesis recicladas Systematization of recycled prostheses* [PhD Thesis]. Universidad de Zaragoza.

Gonzalez, V., Rowson, J., & Yoxall, A. (2015). Development of the Variable Dexterity Test: Construction, reliability and validity. *International Journal of Therapy and Rehabilitation*, *22*(4), 174-180.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Halfacree, G., & Everard, B. (2021). *Get started with MicroPython on Raspberry Pi Pico*. Raspberry Pi Trading Ltd.

Hargrove, L. J., Englehart, K., & Hudgins, B. (2007). A comparison of surface and intramuscular myoelectric signal classification. *IEEE Transactions on Bio-Medical Engineering*, *54*(5), 847-853.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition* (arXiv:1512.03385). arXiv.

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2018). *Densely Connected Convolutional Networks* (arXiv:1608.06993). arXiv.

Huaroto Sevilla, J. J. J. (2019). Diseño de un generador de estímulos mecánicos como interfaz blanda entre prótesis y miembro residual a nivel transradial con capacidad nominal de estimulación entre 0 y 8.5 N a 70 Hz. *Universidad Nacional de Ingeniería*.

Jansen, B. J. (1998). The graphical user interface. *ACM SIGCHI Bulletin*, *30*(2), 22-26.

Kakoty, N. M., Saikia, A., & Hazarika, S. M. (2015). Exploring a family of wavelet transforms for EMG-based grasp recognition. *Signal, Image and Video Processing*, *9*(3), 553-559.

Kawano, T., & Koganezawa, K. (2016). A method of discriminating fingers and wrist action from surface EMG signals for controlling robotic or prosthetic forearm hand. *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 13-18.

Kerdegari, H., Samsudin, K., Rahman Ramli, A., & Mokaram, S. (2013). Development of Wearable Human Fall Detection System Using Multilayer Perceptron Neural Network. *International Journal of Computational Intelligence Systems*, *6*(1), 127-136.

Konrad, P. (2006). *The ABC of EMG*. Noraxon INC. USA.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*.

Kurapa, A., Rathore, D., Edla, D. R., Bablani, A., & Kuppili, V. (2020). A Hybrid Approach for Extracting EMG signals by Filtering EEG Data for IoT Applications for Immobile Persons. *Wireless Personal Communications*, *114*(4), 3081-3101.

Lederman, S. J., & Klatzky, R. L. (1987). Hand movements: A window into haptic object recognition. *Cognitive Psychology*, *19*(3), 342-368.

Leis, A. A., & Trapani, V. C. (2000). *Atlas of Electromyography*. Oxford University Press.

Lu, L., Mao, J., Wang, W., Ding, G., & Zhang, Z. (2020). A Study of Personal Recognition Method Based on EMG Signal. *IEEE Transactions on Biomedical Circuits and Systems*, *14*(4), 681-691.

Lucas Vargas, E. J. (2021). Diseño e implementación de un sistema de asistencia para mejorar la movilidad en personas con discapacidad visual en ambientes externos utilizando redes neuronales convolucionales. *Universidad Nacional de Ingeniería*.

Martinez, W. L. (2011). Graphical user interfaces. *WIREs Computational Statistics*, *3*(2), 119-133.

Miyera, F. (2004). *FILTROS ACTIVOS* (2.ª ed.).

Montanero Fernández, J., & Minuesa Abril, C. (2018). *Estadística básica para Ciencias de la Salud*. Universidad de Extremadura, Servicio de Publicaciones.

Moore, K. L., Dalley, A. F., & Agur, A. M. R. (2013). *Anatomía con Orientación Clínica* (6ta ed.). Lippincott Williams & Wilkins.

Muceli, S., Jiang, N., & Farina, D. (2010). Multichannel surface EMG based estimation of bilateral hand kinematics during movements at multiple degrees of freedom. *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, 6066-6069.

Nahid, N., Rahman, A., & Ahad, M. A. R. (2020a). Deep Learning Based Surface EMG Hand Gesture Classification for Low-Cost Myoelectric Prosthetic Hand. *2020 Joint 9th International Conference on Informatics, Electronics & Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, 1-8.

Naik, G. R. (2012). *Computational Intelligence in Electromyography Analysis: A Perspective on Current Applications and Future Challenges*. BoD – Books on Demand.

Nazmi, N., Abdul Rahman, M. A., Yamamoto, S.-I., Ahmad, S. A., Zamzuri, H., & Mazlan, S. A. (2016). A Review of Classification Techniques of EMG Signals during Isotonic and Isometric Contractions. *Sensors*, *16*(8), Article 8.

Nurpeissova, A., Tursynbekov, T., & Shintemirov, A. (2021). An Open-Source Mechanical Design of ALARIS Hand: A 6-DOF Anthropomorphic Robotic Hand. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 1177-1183.

Oh, D. C., & Jo, Y. U. (2019). EMG-based hand gesture classification by scale average wavelet transform and CNN. *2019 19th International Conference on Control, Automation and Systems (ICCAS)*, 533-538.

Organización Mundial de la Salud. (2011). *Informe mundial sobre la discapacidad 2011*. Organización Mundial de la Salud.

Parajuli, N., Sreenivasan, N., Bifulco, P., Cesarelli, M., Savino, S., Niola, V., Esposito, D., Hamilton, T. J., Naik, G. R., Gunawardana, U., & Gargiulo, G. D. (2019). Real-Time EMG Based Pattern Recognition Control for Hand Prostheses: A Review on Existing Methods, Challenges and Future Implementation. *Sensors*, *19*(20), Article 20.

Peña, C. (2020). *Arduino IDE: Domina la programación y controla la placa*. RedUsers.

Pequera, G. (2015). *Análisis tiempo-frecuencia de la señal de EMG en movimientos explosivos: Estudio de la coordinación en el salto vertical*.

Perotto, A. O. (2011). *ANATOMICAL GUIDE FOR THE ELECTROMYOGRAPHER: The Limbs and Trunk*. Charles C Thomas Publisher.

PERU21, N. (2021, febrero 9). *PUCP crea prótesis de mano que mejorará la calidad de vida de personas con discapacidad | PERU*. Peru21; NOTICIAS PERU21. https://peru21.pe/peru/pucp-crea-protesis-de-mano-que-mejorara-la-calidad-de-vida-de-personas-con-discapacidad-noticia/

Phinyomark, A., N. Khushaba, R., & Scheme, E. (2018). Feature Extraction and Selection for Myoelectric Control Based on Wearable EMG Sensors. *Sensors*, *18*(5), Article 5.

Preece, S. J., Goulermas, J. Y., Kenney, L. P. J., Howard, D., Meijer, K., & Crompton, R. (2009). Activity identification using body-mounted sensors—A review of classification techniques. *Physiological Measurement*, *30*(4), R1.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., & Sohl-Dickstein, J. (2017). *On the Expressive Power of Deep Neural Networks* (arXiv:1606.05336). arXiv.

Rainoldi, A., Melchiorri, G., & Caruso, I. (2004). A method for positioning electrodes during surface EMG recordings in lower limb muscles. *Journal of Neuroscience Methods*, *134*(1), 37-43.

Riaño, C., & Quintero, V. (2010). *Control de una mano virtual usando señales electromiográficas* [Universidad Militar Nueva Granada].

Rocha, C. R., Tonetto, C. P., & Dias, A. (2011). A comparison between the Denavit–Hartenberg and the screw-based methods used in kinematic modeling of robot manipulators. *Robotics and Computer-Integrated Manufacturing*, *27*(4), 723-728.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2019). *MobileNetV2: Inverted Residuals and Linear Bottlenecks* (arXiv:1801.04381). arXiv.

Sarmiento-Ramos, J. L. (2020). Aplicaciones de las redes neuronales y el deep learning a la ingeniería biomédica. *Revista UIS Ingenierías*, *19*(4), Article 4.

Sepúlveda, S., Reyes, P., & Weinstein, A. (2015). *Visualizing physiological signals in real-time*. 182-186.

Shin, S., Kang, M., Zhang, G., Jung, J., & Kim, Y. T. (2022). Lightweight Ensemble Network for Detecting Heart Disease Using ECG Signals. *Applied Sciences*, *12*(7), Article 7.

Soto Bustamante, C. E. (2016). *Exportación dispositivos médicos al Perú*.

Talebi, H., & Milanfar, P. (2021). *Learning to Resize Images for Computer Vision Tasks* (arXiv:2103.09950). arXiv.

Telegenov, K., Tlegenov, Y., & Shintemirov, A. (2015). A Low-Cost Open-Source 3-D-Printed Three-Finger Gripper Platform for Research and Educational Purposes. *IEEE Access*, *3*, 638-647.

Ting, J., Del Vecchio, A., Friedenberg, D., Liu, M., Schoenewald, C., Sarma, D., Collinger, J., Colachis, S., Sharma, G., Farina, D., & Weber, D. J. (2019). A wearable neural interface for detecting and decoding attempted hand movements in a person with tetraplegia. *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 1930-1933.

Too, J., Abdullah, A. R., Saad, N. M., Ali, N. M., & Musa, H. (2018). A Detail Study of Wavelet Families for EMG Pattern Recognition. *International Journal of Electrical and Computer Engineering (IJECE)*, *8*(6), Article 6.

Turnip, A., Kusumandari, D. E., Arson, G. W. G., & Setiadikarunia, D. (2021). Electric Wheelchair Controlled-Based EMG with Backpropagation Neural Network Classifier. En E. Joelianto, A. Turnip, & A. Widyotriatmo (Eds.), *Cyber Physical, Computer and Automation System: A Study of New Technologies* (pp. 149-155). Springer.

Vergara, M., Cabedo, J. S., Cervantes, P. R., & González, A. P. (2012). *Resultados de un trabajo de campo sobre agarres utilizados en tareas cotidianas utilizados en tareas cotidianas*.

World plugs | IEC. (s. f.). Recuperado 29 de diciembre de 2022, de https://www.iec.ch/world-plugs

Zhang, X., & Zhou, P. (2013). Filtering of surface EMG using ensemble empirical mode decomposition. *Medical Engineering & Physics*, *35*(4), 537-542.

Zhang, Z., Yang, K., Qian, J., & Zhang, L. (2019). Real-Time Surface EMG Pattern Recognition for Hand Gestures Based on an Artificial Neural Network. *Sensors (Basel, Switzerland)*, *19*(14), 3170.

Zhu, L., Mao, G., Su, H., Zhou, Z., Li, W., Lü, X., & Wang, Z. (2021). A Wearable, High-Resolution, and Wireless System for Multichannel Surface Electromyography Detection. *IEEE Sensors Journal*, *21*(8), 9937-9948.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2021). A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, *109*(1), 43-76.

Zschorlich, V. R. (1989). Digital filtering of EMG-signals. *Electromyography and Clinical Neurophysiology*, *29*(2), 81-86.

ANEXOS

ANEXOS A

CÓDIGOS FUENTE EN ARDUINO Y PYTHON

A.1. CÓDIGO DE ADQUISICIÓN PARA LA RASPBERRY PI PICO

```
// ADQUISICIÓN DE 3 SENSORES EMG A 1000 Hz
// Librerias
#include "pico/stdlib.h"
#define SERIAL_TX_BUFFER_SIZE 4096
#define TIMING_DEBUG 1
// Primer sensor
const int Sensor1InputPin = 28;
// Segundo sensor
const int Sensor2InputPin = 27;
// Tercer sensor
const int Sensor3InputPin = 26;
unsigned long timeStamp;
unsigned long timeBudget;

void setup() {
    // open serial Computadora
    Serial.begin(230400);
    analogReadResolution(10);
}
void loop() {
    timeStamp = micros();
    // Leer el valor analógico
    int Value1 = analogRead(Sensor1InputPin);
    int Value2 = analogRead(Sensor2InputPin);
    int Value3 = analogRead(Sensor3InputPin);
    String Value1S = (String) Value1;
    String Value2S = (String) Value2;
    String Value3S = (String) Value3;
    // Tiempo de adquisición
    timeStamp = micros() - timeStamp;
    if (TIMING_DEBUG) {
        // Envío de los datos de los 3 canales
        Serial.print(Value1S + ',' + Value2S + ',' +Value3S + "\r\n" );
    }
    // Mantener una frecuencia de 1000 Hz
    delayMicroseconds(1000 - timeStamp);
}
```

A.2. FUNCIONES PARA LA GUI (INCLUYENDO CLASIFICACIÓN)

```
# Importamos librerias principales --------------
import numpy as np
import sys
import serial, time
import cv2
import threading
import argparse
import datetime
from PIL import Image
import os
from multiprocessing.sharedctypes import RawArray
from random import randint
from tqdm import tqdm
import pandas as pd
import socket
from joblib import dump, load
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from skimage.transform import resize
from sklearn.preprocessing import StandardScaler
# Importamos librerias de DL--------------------
import torch
from torch.nn import Linear, CrossEntropyLoss
from torch.optim import Adam
from torch.utils.data import DataLoader
import torchvision
from torchvision.datasets import ImageFolder
from torchvision.models import mobilenet_v2
from torchvision.transforms import transforms
import torch.nn as nn
# Importamos librerias PyQt5 - GUI --------------
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5 import QtGui
from PyQt5.QtWidgets import QWidget, QApplication, QLabel,
from PyQt5.QtWidgets import QMainWindow, QPushButton, QProgressBar
from PyQt5.QtWidgets import QVBoxLayout, QSplitter, QFrame
from PyQt5.QtGui import QPixmap
from PyQt5.QtCore import pyqtSignal, pyqtSlot, Qt, QThread
from PyQt5.QtGui import QPainter
from pyqtgraph import PlotWidget, plot
import pyqtgraph as pg
from PyQt5 import QtWidgets, QtCore
# Importamos librerias de gráfica --------------
import matplotlib.pyplot as plt
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from matplotlib import cm
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D

```

```
# Importamos librerias de Señales -------------
import scipy
from scipy import signal
from scipy.signal import freqz
from scipy.signal import butter, lfilter
import pywt

# ESTANDARIZACIÓN  ------------------------------
def standard_EMG(data,scaler_EMG):
    data = [data]
    data = np.transpose(data)
    data = scaler_EMG.transform(data)
    standard = np.transpose(data)
    standard = standard[0]
    return standard
# FILTRO PASABANDA  ------------------------------
def butter_bandpass(lowcut, highcut, fs, order):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a
#Funcion para el bandpass
def butter_bandpass_filter(data, lowcut, highcut, fs, order):
    b, a = butter_bandpass(lowcut, highcut, fs, order)
    y = signal.filtfilt(b, a, data)
    return y
def band_pass(p):
    lowcut = 10.0
    highcut = 499.0
    samplingFrequency = 1000
    p_banda = butter_bandpass_filter(p, lowcut, highcut,
                                     samplingFrequency, 4)
    return p_banda
# FILTRO NOTCH  ------------------------------
def notch_filter(p):
    notch_freq = 60  # Frequencia a señal remover (Hz)
    quality_factor = 2  # factor de calidad
    samplingFrequency= 1000 # 1k Hz
    b_notch, a_notch = signal.iirnotch(notch_freq, quality_factor,
                                       samplingFrequency)
    freq, h = signal.freqz(b_notch, a_notch, fs=2*np.pi)
    # Apply notch filter to the noisy signal using signal.filtfilt
    p_notch = signal.filtfilt(b_notch, a_notch, p)
    return p_notch
def preprocess (p,signal,scaler_EMG):
    # Primero su estandarización
    p = standard_EMG(p,scaler_EMG)
    # Luego el filtro pasabanda
    p = band_pass(p)
    # Luego el filtro notch
    p = notch_filter(p)
    return p
```

```
105 # TRANFORMADA WAVELET CONTINUA ----------------------------
106 def create_cwt_images(X, n_scales, rescale_size, wavelet_name,
107                        S1, S2, S3):
108     n_samples = X.shape[0]
109     n_signals = X.shape[2]
110     # range of scales from 1 to n_scales
111     scales = np.arange(1, n_scales + 1)
112     # pre allocate array
113     X_cwt = np.ndarray(shape=(n_samples, rescale_size,rescale_size,
114                               n_signals), dtype = 'float32')
115     for sample in range(n_samples):
116         for signal in range(n_signals):
117             serie = X[sample, :, signal]
118             # Procesamiento
119             if signal == 0:
120                 o = preprocess(serie,signal,S1)
121             elif signal == 1:
122                 o = preprocess(serie,signal,S2)
123             elif signal == 2:
124                 o = preprocess(serie,signal,S3)
125             # CWT
126             coeffs, freqs = pywt.cwt(o, scales, wavelet_name)
127             # resize the 2D cwt coeffs
128             rescale_coeffs=resize(coeffs, (rescale_size, rescale_size),
129                                   mode = 'constant')
130             X_cwt[sample,:,:,signal] = rescale_coeffs
131     return X_cwt
132
133 def processing_function(ReHand, PlPush, AdThumb, LaPinch, ThIndex):
134     ##### Datos de la adquisición
135     frecuencia = 1000  # Hz
136     dt = 1/frecuencia # Periodo 0.001
137     Datos = 60000 # numero de datos
138     Ttotal = int(Datos/frecuencia) # Tiempo total 60 seg
139     Ventana = 0.2 # Ventana de señal 200
140     time_s  = np.arange(0, Ventana, dt); # Tiempo
141     part = int(Ttotal/Ventana) # 200 partes
142     win = int(frecuencia*Ventana) # 200 muestras por ventana
143     Nsensor = 3 # número de sensores
144     Nmov = 5 # número de movimientos manuales
145     #### Datos de los movimientos manuales
146     dato1_csv = ReHand + ".csv"
147     dato2_csv = PlPush + ".csv"
148     dato3_csv = AdThumb + ".csv"
149     dato4_csv = LaPinch + ".csv"
150     dato5_csv = ThIndex + ".csv"
151     datos_csv = [dato1_csv, dato2_csv, dato3_csv, dato4_csv, dato5_csv]
152     EMG = ["EMG_1", "EMG_2","EMG_3"]
153     # Unimos las señales
154     XMove = np.zeros([part*Nmov,win,Nsensor])
155     YMove = np.zeros([part*Nmov,1])
156     for j in range(Nmov):
157         df = pd.Cov = pd.read_csv(datos_csv[j], names = EMG)
```

```
158         first = df["EMG_1"]
159         second = df["EMG_2"]
160         third = df["EMG_3"]
161         # Datos del csv a numpy
162         first_np = first.to_numpy()
163         second_np = second.to_numpy()
164         third_np = third.to_numpy()
165         # Entrada (XMove) y label(YMove)
166         output_emg1 = np.zeros([part,win])
167         output_emg2 = np.zeros([part,win])
168         output_emg3 = np.zeros([part,win])
169         output_emg1[0:part]=[first_np[k:k + win]
170                              for k in range(0, len(first_np), win)]
171         output_emg2[0:part]=[second_np[k:k + win]
172                              for k in range(0, len(second_np), win)]
173         output_emg3[0:part]=[third_np[k:k + win]
174                              for k in range(0, len(third_np), win)]
175         output = np.zeros([part,win,Nsensor])
176         output[:part,:win,0] = output_emg1
177         output[:part,:win,1] = output_emg2
178         output[:part,:win,2] = output_emg3
179         label = j*np.ones([part,1])
180         XMove[(j*part):((j+1)*part)] = output
181         YMove[(j*part):((j+1)*part)] = label
182     # ESTANDARIZACIÓN  ------------------------------
183     datos_EMG1 = XMove[:,:win,0]
184     datos_EMG1 = datos_EMG1.flatten()
185     datos_EMG1 = np.transpose([datos_EMG1])
186     S1 = StandardScaler().fit(datos_EMG1)
187     dump(S1, 'S1.bin', compress=True)
188     datos_EMG2 = XMove[:,:win,1]
189     datos_EMG2 = datos_EMG2.flatten()
190     datos_EMG2 = np.transpose([datos_EMG2])
191     S2 = StandardScaler().fit(datos_EMG2)
192     dump(S2, 'S2.bin', compress=True)
193     datos_EMG3 = XMove[:,:win,2]
194     datos_EMG3 = datos_EMG3.flatten()
195     datos_EMG3 = np.transpose([datos_EMG3])
196     S3 = StandardScaler().fit(datos_EMG3)
197     dump(S3, 'S3.bin', compress=True)
198     ## Mezclar y dividir en Entrenamiento y Testeo
199     X_total, y_total = shuffle(XMove, YMove, random_state=10)
200     split_total = train_test_split(X_total, y_total, test_size = 0.3)
201     X_train, X_test, y_train, y_test = split_total
202     # Tranformada CWT para convertir las señales a escalogramas
203     rescale_size = 224
204     n_scales = 15
205     wave = "mexh"
206     X_train_cwt = create_cwt_images(X_train, n_scales,
207                                     rescale_size, wave, S1, S2, S3)
208     print(f"shapes (n_samples, x_img, y_img, z_img) of X_train_cwt:
209           {X_train_cwt.shape}")
210     X_test_cwt = create_cwt_images(X_test, n_scales, rescale_size,
```

```
                                        wave, S1, S2, S3)
    print(f"shapes (n_samples, x_img, y_img, z_img) of X_test_cwt:
          {X_test_cwt.shape}")
    # Eliminamos los archivos de las carpetas
    for i in range(5):
        dir_train = 'train/'+ str(i)
        # Borrar archivos del "train"
        for f in os.listdir(dir_train):
            os.remove(os.path.join(dir_train, f))
        dir_test = 'test/'+ str(i)
        # Borrar archivos del "test"
        for f in os.listdir(dir_test):
            os.remove(os.path.join(dir_test, f))
    # Convertir el numpy en imagen y guardar las imágenes
    for i in range(len(X_train_cwt)):
        label = int(y_train[i][0])
        im = Image.fromarray(((X_train_cwt[i]+10)
                              *255/20).astype(np.uint8))
        im.save("train/"+str(label)+"/"+str(i)+"."+str(label)+".jpg")
    for i in range(len(X_test_cwt)):
        label = int(y_test[i][0])
        im = Image.fromarray(((X_test_cwt[i]+10)
                              *255/20).astype(np.uint8))
        #im = Image.fromarray((X_test_cwt[i]).astype(np.uint8))
        im.save("test/"+str(label)+"/"+str(i)+"."+str(label)+".jpg")

def training_function():
    # Device
    device = torch.device("cuda"
                          if torch.cuda.is_available() else "cpu")
    # Data Transformer
    tfm = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],
                             [0.229, 0.224, 0.225])
    ])
    # Configuración del modelo
    mobilenet_v2model = torch.jit.load('mobilenet_v2model.pt')
    torch.manual_seed(42)
    # Optimiser
    optimiser = Adam(mobilenet_v2model.parameters(), lr=3e-4,
                     weight_decay=0.0001)
    # Loss Function
    loss_fn = CrossEntropyLoss()
    # Crear Dataset
    TRAIN_ROOT = "train"
    TEST_ROOT = "test"
    train_ds = ImageFolder(TRAIN_ROOT, transform=tfm)
    test_ds = ImageFolder(TEST_ROOT, transform=tfm)
    LEN_TRAIN = len(train_ds)
    LEN_TEST = len(test_ds)
    # Data Loader
    train_loader = DataLoader(train_ds, batch_size = 30, shuffle=True)
```

```
    test_loader = DataLoader(test_ds, batch_size = 30, shuffle = True)
    train_losses=[]
    test_losses=[]
    train_correct=[]
    test_correct=[]
    epochs = 1
    for epoch in range(epochs):
        start = time.time()
        tr_acc = 0
        test_acc = 0
        # Train
        mobilenet_v2model.train()
        with tqdm(train_loader, unit="batch") as tepoch:
            for xtrain, ytrain in tepoch:
                optimiser.zero_grad()
                xtrain = xtrain.to(device)
                train_prob = mobilenet_v2model(xtrain)
                train_prob = train_prob.cpu()
                loss_train = loss_fn(train_prob, ytrain)
                loss_train.backward()
                optimiser.step()
                train_pred = torch.max(train_prob, 1).indices
                tr_acc += int(torch.sum(train_pred == ytrain))
            ep_tr_acc = tr_acc / LEN_TRAIN
            # Train
            loss_train = loss_train.detach().numpy()
            train_losses.append(loss_train)
            train_correct.append(ep_tr_acc)
        # Evaluate
        mobilenet_v2model.eval()
        with torch.no_grad():
            for xtest, ytest in test_loader:
                xtest = xtest.to(device)
                test_prob = mobilenet_v2model(xtest)
                test_prob = test_prob.cpu()
                loss_test = loss_fn(test_prob, ytest)
                test_pred = torch.max(test_prob,1).indices
                test_acc += int(torch.sum(test_pred == ytest))
            ep_test_acc = test_acc / LEN_TEST
            # Test
            loss_test = loss_test.detach().numpy()
            test_losses.append(loss_test)
            test_correct.append(ep_test_acc)
        end = time.time()
        duration = (end - start) / 60
        print(f"Epoch: {epoch}, Time: {duration}, Loss_train:
              {loss_train}\nTrain_acc: {ep_tr_acc}, Test_acc:
              {ep_test_acc}, Loss_test: {loss_test}")
    # Guardar modelo
    model_scripted = torch.jit.script(mobilenet_v2model)
    model_scripted.save('mobilenet_v2model.pt') # Save
    return str(round(ep_test_acc, 4)), str(round(float(loss_test), 4))
```

```
317 def create_cwt_image(X, n_scales, rescale_size,
318                         wavelet_name, S1, S2, S3):
319     # n_samples = X.shape[0] 1
320     n_signals = X.shape[1]
321     # range of scales from 1 to n_scales
322     scales = np.arange(1, n_scales + 1)
323     # pre allocate array
324     X_cwt = np.ndarray(shape=(rescale_size, rescale_size, n_signals),
325                        dtype = 'float32')
326     for signal in range(n_signals):
327         # Each signal
328         serie = X[:, signal]
329         # Procesamiento
330         if signal == 0:
331             o = preprocess(serie,signal,S1)
332         elif signal == 1:
333             o = preprocess(serie,signal,S2)
334         elif signal == 2:
335             o = preprocess(serie,signal,S3)
336         # continuous wavelet transform
337         coeffs, freqs = pywt.cwt(o, scales, wavelet_name)
338         # resize the 2D cwt coeffs
339         rescale_coeffs = resize(coeffs, (rescale_size, rescale_size),
340                                 mode = 'constant')
341         X_cwt[:,:,signal] = rescale_coeffs
342     return X_cwt
343 # Procesamiento
344 def img_process(signal_200_3,S1,S2,S3):
345     rescale_size = 224
346     n_scales = 15
347     wave = "mexh"
348     # signal to image CWT
349     img_3 = create_cwt_image(signal_200_3, n_scales, rescale_size,
350                              wave,S1,S2,S3)
351     # CWT to jpg
352     Actual_img = Image.fromarray(((img_3+10)*255/20).astype(np.uint8))
353     #Actual_img = Image.fromarray(img_3.astype(np.uint8))
354     Actual_img.save("actual/actual.jpg")
355 # Clasificación
356 def img_classication(model_actual):
357     model_actual.eval()
358     img_as_img = Image.open("actual/actual.jpg")
359     tfm = transforms.Compose([
360         transforms.ToTensor(),
361         transforms.Normalize([0.485, 0.456, 0.406],
362                              [0.229, 0.224, 0.225])
363     ])
364     Actual_tensor = tfm(img_as_img)
365     with torch.no_grad():
366         new_pred =
367               model_actual(Actual_tensor.view(1,3,224,224)).argmax()
368     return new_pred.item()
```

A.3. CÓDIGO FUENTE DE LA GUI

```
## código de la GUI - cliente 1 ----------------------------------
nombreX = "0"
nombreY = "0"
nombreZ = "0"
# Bandera para guardar datos
FLAG_SAVE_1 = 0
FLAG_SAVE_2 = 0
FLAG_SAVE_3 = 0
FLAG_SAVE_4 = 0
FLAG_SAVE_5 = 0
# Bandera de proceso
FLAG_PROCESSING = 0
FLAG_TRAINING = 0
FLAG_PREDICTING = 0
FLAG_TIME = 0
estadoPro = 0
estadoProAnt = 0
# Datos de adquisición ------------------------------------------
fss = 1000 # 1k Hz
Nvent = 200 # 300 datos por ventana
dt = 1/fss # Periodo 0.001 seg
Pvent= Nvent*dt # periodo de ventana 0.2 seg
Ttotal = 60 # 1 minuto
Datos = Ttotal*fss # 60k datos en 1 minuto
part = int(Ttotal/Pvent) # 120 partes
Nsensor = 3 # número de sensores
# Cantidad de tiempo para salvar
a1 = int(Datos)
a2 = int(Datos)
a3 = int(Datos)
# Cantidad de tiempo para hacer la imagen  0<=b<500
b = 0
estadoAdq = 0
estadoAdqAnt = 0
# El buffer para la imagen a clasificar ---------------------------
output = np.zeros([Nvent,Nsensor])
# Conexión serial Rasbperry Pico ----------------------------------
pico = serial.Serial('COM9', 230400)
time.sleep(2)
# Panel de control ----------------
namemove1 = " "
namemove2 = " "
namemove3 = " "
namemove4 = " "
namemove5 = " "
acc = " "
loss = " "
# Valores de la normalización
S1 = load('S1.bin')
S2 = load('S2.bin')
S3 = load('S3.bin')
```

```
52 # Fe modelo
53 model = torch.jit.load('mobilenet_v2model.pt')
54 host = '127.0.0.1'
55 port = 1233
56 ClientSocket = socket.socket()
57 print('Waiting for connection')
58 try:
59     ClientSocket.connect((host, port))
60 except socket.error as e:
61     print(str(e))
62 Response = ClientSocket.recv(1024)
63 iii = 0
64 ## Creamos la clase del hilo de adquisición -------------------------
65 class AdqThread(QThread):
66     # Funcion de salida del hilo
67     adq_signals = pyqtSignal(str,str,str,np.ndarray)
68     # Funcion de inicio del hilo
69     def __init__(self):
70         super().__init__()
71         self._run_flag = True
72     def __del__(self):
73         self.wait()
74     # Funcion principal
75     def run(self):
76         nombreX = "0"
77         nombreY = "0"
78         nombreZ = "0"
79         # Corre constantemente
80         while self._run_flag:
81             # Variables globales de la señal actual
82             global b, a1, a2, a3, a4, a5, Datos, estadoAdq
83             global Nvent, output, hola
84             global FLAG_SAVE_1, FLAG_SAVE_2, FLAG_SAVE_3
85             global FLAG_SAVE_4, FLAG_SAVE_5
86             global namemove1,namemove2,namemove3,namemove4,namemove5
87             # Si la comunicación esta abierta
88             if (pico.isOpen()):
89                 dt=pico.readline() #Esto se recibe en bytes
90                 try:
91                     Dt_s = dt.decode('UTF-8') #de Byte a String
92                     trozos = Dt_s.split(',')
93                 except:
94                     print("error")
95                 if b < Nvent:
96                     # Primera señal
97                     try:
98                         nombreX = trozos[0]
99                         output[b,0] = int(nombreX)
100                    except:
101                        print("error")
102                    # Segunda señal
103                    try:
104                        nombreY = trozos[1]
```

```
105                         output[b,1] = int(nombreY)
106                     except:
107                         print("error")
108                     # Tercera señal
109                     try:
110                         nombreZ = trozos[2]
111                         output[b,2] = int(nombreZ)
112                     except:
113                         print("error")
114                     # Cuando se llena la ventana
115                     b+=1
116                     if b == Nvent:
117                         b = 0
118                 # Guardar en un csv el move 1
119                 if (FLAG_SAVE_1 == 1):
120                     if (a1 == Datos):
121                         estadoAdq = 1
122                         file1 = open(namemove1+".csv","w")
123                     if a1>0:# program logic
124                         file1.write(str(int(nombreX))
125                                     +","+str(int(nombreY))
126                                     +","+str(int(nombreZ))+"\n")
127                         file1.flush()# internal buffer is flushed
128                         a1-=1
129                         if a1 == 0:
130                             estadoAdq = 2
131                             file1.close()
132                             FLAG_SAVE_1 = 0
133                 # Guardar en un csv el move 2
134                 if (FLAG_SAVE_2 == 1):
135                     if (a2 == Datos):
136                         estadoAdq = 3
137                         file2 = open(namemove2+".csv","w")
138                     if a2>0:# program logic
139                         file2.write(str(int(nombreX))
140                                     +","+str(int(nombreY))
141                                     +","+str(int(nombreZ))+"\n")
142                         file2.flush()# internal buffer is flushed
143                         a2-=1
144                         if a2 == 0:
145                             estadoAdq = 4
146                             file2.close()
147                             FLAG_SAVE_2 = 0
148                  # Guardar en un csv el move 3
149                 if (FLAG_SAVE_3 == 1):
150                     if (a3 == Datos):
151                         estadoAdq = 5
152                         file3 = open(namemove3+".csv","w")
153                     if a3>0:# program logic
154                         file3.write(str(int(nombreX))
155                                     +","+str(int(nombreY))
156                                     +","+str(int(nombreZ))+"\n")
157                         file3.flush()# internal buffer is flushed
```

```
158                         a3-=1
159                         if a3 == 0:
160                             estadoAdq = 6
161                             file3.close()
162                             FLAG_SAVE_3 = 0
163                 # Guardar en un csv el move 4
164                 if (FLAG_SAVE_4 == 1):
165                     if (a4 == Datos):
166                         estadoAdq = 7
167                         file4 = open(namemove4+".csv","w")
168                     if a4>0:# program logic
169                         file4.write(str(int(nombreX))
170                                     +","+str(int(nombreY))
171                                     +","+str(int(nombreZ))+"\n")
172                         file4.flush()# internal buffer is flushed
173                         a4-=1
174                         if a4 == 0:
175                             estadoAdq = 8
176                             file4.close()
177                             FLAG_SAVE_4 = 0
178                 # Guardar en un csv el move 5
179                 if (FLAG_SAVE_5 == 1):
180                     if (a5 == Datos):
181                         estadoAdq = 9
182                         file5 = open(namemove5+".csv","w")
183                     if a5>0:# program logic
184                         file5.write(str(int(nombreX))
185                                     +","+str(int(nombreY))
186                                     +","+str(int(nombreZ))+"\n")
187                         file5.flush()# internal buffer is flushed
188                         a5-=1
189                         if a5 == 0:
190                             estadoAdq = 10
191                             file5.close()
192                             FLAG_SAVE_5 = 0
193                 # Funcion de envio de las señales
194                 self.adq_signals.emit(nombreX,nombreY,nombreZ,output)
195     def stop(self):
196         # Envia una bandera para parar el hilo
197         self._run_flag = False
198         self.wait()
199 ## Creamos la clase del hilo de predicción -------------------------
200 class PreThread(QThread):
201     # Funcion de salida del hilo
202     pre_signals = pyqtSignal(str)
203     # Funcion de inicio del hilo
204     def __init__(self):
205         super().__init__()
206         self._run_flag = True
207     def __del__(self):
208         self.wait()
209     # Funcion principal
210     def run(self):
```

```
211         # Corre constantemente
212         label_p = "Sin predecir"
213         while self._run_flag:
214             # Matriz actual
215             global output
216             #global model
217             global Nvent, Nsensor
218             global estadoPro
219             global namemove1,namemove2,namemove3,namemove4,namemove5
220             #FLAGS
221             global FLAG_PROCESSING, FLAG_TRAINING
222             global FLAG_PREDICTING, FLAG_TIME
223             global acc, loss
224             global S1, S2, S3, model
225             # Cuando se presiona el botón de procesamiento
226             if (FLAG_PROCESSING == 1):
227                 # Procesando
228                 estadoPro = 1
229                 processing_function(namemove1,
230                                     namemove2, namemove3,
231                                     namemove4, namemove5)
232                 # Finalizó procesamiento
233                 estadoPro = 2
234                 FLAG_PROCESSING = 0
235             # Cuando se presiona el botón de entrenamiento
236             if (FLAG_TRAINING == 1):
237                 # Procesando
238                 estadoPro = 3
239                 acc, loss = training_function()
240                 # Finalizó procesamiento
241                 estadoPro = 4
242                 FLAG_TRAINING = 0
243             # Cuando se presiona el botón de predicción
244             if (FLAG_PREDICTING == 1):
245                 if(FLAG_TIME == 1):
246                     start_time = time.time()
247                     file_predict = open(namemove1+"PRED"+".csv","w")
248                     FLAG_TIME = 2
249                 # presionado por segunda vez
250                 if(FLAG_TIME == 3):
251                     file_predict.close()
252                     print("cerrar predicción")
253                     FLAG_TIME = 4
254                 # Procesando
255                 estadoPro = 5
256                 # Procesar la ventana actual
257                 img_process(output, S1, S2, S3)
258                 # Clasificación de la ventana actual int
259                 clasif = img_classication(model)
260                 if clasif == 0:
261                     label_p = " Relaxed Hand"
262                 elif clasif == 1:
263                     label_p = "Platform Push"
```

```
264                 elif clasif == 2:
265                     label_p = "Adducted Thumb"
266                 elif clasif == 3:
267                     label_p = "Lateral Pinch"
268                 elif clasif == 4:
269                     label_p = "Thumb Index"
270                 if (FLAG_TIME == 2):
271                     t_time = round((time.time()-start_time) * 1000)
272                     file_predict.write(str(t_time)
273                                        +","+label_p+"\n")
274                     file_predict.flush()# internal buffer is flushed
275                 ClientSocket.send(str.encode(str(clasif)))
276                 Response = ClientSocket.recv(1024)
277                 #print(Response.decode('utf-8'))
278             if (FLAG_PREDICTING == 0):
279                 time.sleep(0.3)
280             self.pre_signals.emit(label_p)
281     def stop(self):
282         # Envia una bandera para parar el hilo
283         self._run_flag = False
284         self.wait()
285 # Clase para label con su valor ---------------------------------
286 class LabelledName(QWidget):
287     def __init__(self, title, initial_value, layout):
288         QWidget.__init__(self)
289         #layout = QVBoxLayout()
290         #self.setLayout(layout)
291         # Label del título del sensor
292         self.label = QLabel()
293         self.label.setText(title)
294         self.label.setFont(QFont("Arial",weight=QFont.Bold))
295         self.label.setAlignment(Qt.AlignCenter)
296         layout.addWidget(self.label)
297         # Label del valor del sensor
298         self.label_value = QLabel()
299         self.label_value.setAlignment(Qt.AlignCenter)
300         #if initial_value != None:
301         self.label_value.setText(initial_value)
302         layout.addWidget(self.label_value)
303         #layout.addStretch()
304     def setValue(self, value):
305         self.label_value.setText(value)
306     def getValue(self):
307         return int(self.label_value.text())
308 # Clase para predict----------------------------------------------
309 class LabelledName2(QWidget):
310     def __init__(self, title, initial_value):
311         QWidget.__init__(self)
312         layout = QVBoxLayout()
313         self.setLayout(layout)
314         # Label del título del sensor
315         self.label = QLabel()
316         self.label.setText(title)
```

```
317         self.label.setFont(QFont("Arial",weight=QFont.Bold))
318         self.label.setAlignment(Qt.AlignCenter)
319         layout.addWidget(self.label)
320         # Label del valor del sensor
321         self.label_value = QLabel()
322         self.label_value.setAlignment(Qt.AlignCenter)
323         #if initial_value != None:
324         self.label_value.setText(initial_value)
325         layout.addWidget(self.label_value)
326         #layout.addStretch()
327     def setValue(self, value):
328         self.label_value.setText(value)
329     def getValue(self):
330         return int(self.label_value.text())
331 # Clase para la adquisición de cada movimiento manual--------------------
332 class hand_move(QWidget):
333     def __init__(self, hand_name,
334                  default_name, abrev_name, functionbutton):
335         QWidget.__init__(self)
336         vlayout = QVBoxLayout()
337         self.setLayout(vlayout)
338         # Label del movimiento manual
339         self.label_name = QLabel()
340         self.label_name.setText(hand_name)
341         self.label_name.setFont(QFont("Arial",9))
342         self.label_name.setStyleSheet("font-weight: bold")
343         vlayout.addWidget(self.label_name)
344         # Horizontal box
345         hlayout = QHBoxLayout()
346         vlayout.addLayout(hlayout)
347         # Label del text edit
348         self.lineEdit = QLineEdit(self)
349         self.lineEdit.setFixedWidth(200)
350         self.lineEdit.setValidator(QIntValidator())
351         self.lineEdit.setAlignment(Qt.AlignCenter)
352         self.lineEdit.setText(default_name)
353         hlayout.addWidget(self.lineEdit)
354         # Button
355         self.button = QPushButton(abrev_name)
356         self.button.clicked.connect(functionbutton)
357         hlayout.addWidget(self.button)
358         # Label de estado
359         self.label_state = QLabel()
360         self.label_state.setText("Sin adquirir")
361         self.label_state.setFont(QFont("Arial", 8))
362         self.label_state.setAlignment(Qt.AlignCenter) #centrar
363         vlayout.addWidget(self.label_state)
364     def getValue(self):
365         return self.lineEdit.text()
366     def setState(self, value):
367         self.label_state.setText(value)
368
369
```

```
370 # Clase para label, button y label ---------------------------------
371 class system process(QWidget):
372     def __init__(self,
373                  process_title, process,
374                  process_state, functionbutton):
375         QWidget.__init__(self)
376         vlayout = QVBoxLayout()
377         self.setLayout(vlayout)
378         # Label del proceso
379         self.label_name = QLabel()
380         self.label_name.setText(process_title)
381         self.label_name.setFont(QFont("Arial",10))
382         self.label_name.setStyleSheet("font-weight: bold")
383         self.label_name.setAlignment(Qt.AlignCenter) #centrar
384         vlayout.addWidget(self.label_name)
385         # Horizontal box
386         hlayout = QHBoxLayout()
387         vlayout.addLayout(hlayout)
388         # Button
389         self.button = QPushButton(process)
390         self.button.clicked.connect(functionbutton)
391         hlayout.addWidget(self.button)
392         # Label de estado de adquisición
393         self.label_state = QLabel()
394         self.label_state.setText(process_state)
395         self.label_state.setFont(QFont("Arial",8))
396         self.label_state.setAlignment(Qt.AlignCenter) #centrar
397         hlayout.addWidget(self.label_state)
398     def setState(self, value):
399         self.label_state.setText(value)
400
401 # Programa principal del GUI -------------------------------------
402 class Demo(QDialog):
403     def __init__(self, parent=None):
404         # Función inicial del GUI
405         QDialog.__init__(self, parent)
406         # Configuración del aspecto del GUI
407         self.setWindowFlags(Qt.WindowStaysOnTopHint)
408         self.setWindowTitle("Program GUI")
409         self.setFixedSize(800, 800)
410         # Box principal en vertical ------------------------------
411         vmainlayout = QVBoxLayout()
412         self.setLayout(vmainlayout)
413         # Label del título
414         self.labelT = QLabel()
415         self.labelT.setText('GUI DEL PROYECTO')
416         #self.label.setFixedWidth(400)
417         self.labelT.setFont(QFont("Arial",14))
418         self.labelT.setStyleSheet("font-weight: bold")
419         self.labelT.setAlignment(Qt.AlignCenter) #centrar
420         vmainlayout.addWidget(self.labelT)
421         # Box principal en horizontal ----------------------------
422         hmainlayout = QHBoxLayout()
```

```
423         vmainlayout.addLayout(hmainlayout)
424         # Box secundario izquierda ------------------------------
425         vizqlayout = QVBoxLayout()
426         hmainlayout.addLayout(vizqlayout)
427         # Box secundario derecha -------------------------------
428         vderlayout = QVBoxLayout()
429         hmainlayout.addLayout(vderlayout)
430         # Label de comunicación serial
431         self.labelCon = QLabel()
432         self.labelCon.setText('Comunicación serial')
433         self.labelCon.setFont(QFont("Arial",10))
434         self.labelCon.setStyleSheet("font-weight: bold")
435         self.labelCon.setAlignment(Qt.AlignCenter) #centrar
436         vizqlayout.addWidget(self.labelCon)
437         # Box para los botones de comunicación serial
438         hconlayout = QHBoxLayout()
439         vizqlayout.addLayout(hconlayout)
440         self.addButtonReconnect(hconlayout)
441         self.addButtonDisconnect(hconlayout)
442         # Label de botones de adquisición
443         self.labelAdq = QLabel()
444         self.labelAdq.setText('Adquisición de las señales EMG')
445         self.labelAdq.setFont(QFont("Arial",10))
446         self.labelAdq.setStyleSheet("font-weight: bold")
447         self.labelAdq.setAlignment(Qt.AlignCenter) #centrar
448         vizqlayout.addWidget(self.labelAdq)
449         # Movimiento 1
450         self.move1 = hand_move("Mano relajada", "RelaxedHand",
451                                "ReHand", self.buttonfM1)
452         vizqlayout.addWidget(self.move1)
453         # Movimiento 2
454         self.move2 = hand_move("Empuje de plataforma","PlatformPush",
455                                "PlPush", self.buttonfM2)
456         vizqlayout.addWidget(self.move2)
457         # Movimiento 3
458         self.move3 = hand_move("Pulgar aducido", "AdductedThumb",
459                                "AdThumb", self.buttonfM3)
460         vizqlayout.addWidget(self.move3)
461         # Movimiento 4
462         self.move4 = hand_move("Pinza lateral", "LateralPinch",
463                                "LaPinch", self.buttonfM4)
464         vizqlayout.addWidget(self.move4)
465         # Movimiento 5
466         self.move5 = hand_move("Pinza pulgar-índice", "ThumbIndex",
467                                "ThIndex", self.buttonfM5)
468         vizqlayout.addWidget(self.move5)
469         # Procesamiento de las señales -------------------------
470         self.processing=system_process("Procesamiento de las señales",
471                             "Procesar", "Sin procesar",
472                                        self.buttonProcess)
473         vizqlayout.addWidget(self.processing)
474         # Entrenamiento del modelo -----------------------------
475         self.training=system_process("Entrenamiento del modelo",
```

```
476                         "Entrenar","Sin entrenar",self.buttonTrain)
477         vizqlayout.addWidget(self.training)
478         # Label de precisión
479         self.labelPre = QLabel()
480         self.labelPre.setText("Precisión: 0.0000        0.0000")
481         self.labelPre.setFont(QFont("Arial",9))
482         self.labelPre.setAlignment(Qt.AlignCenter) #centrar
483         vizqlayout.addWidget(self.labelPre)
484         # Clasificación de los movimiento manuales-----------------
485         self.prediction = system_process("Predicción del movimiento",
486                         "Predecir", "---", self.buttonPredict)
487         vizqlayout.addWidget(self.prediction)
488         # Gráfica del sensor 1
489         f_layout = QVBoxLayout()
490         self.first = LabelledName('Primer canal', '0', f_layout)
491         self.graphWidget1 = pg.PlotWidget()
492         self.x1 = list(range(200))  # 200 time points
493         self.y1 = [randint(0,100) for _ in range(200)]
494         self.graphWidget1.setBackground('w')
495         pen1 = pg.mkPen(color=(255, 0, 0))
496         self.data_line1 =  self.graphWidget1.plot(self.x1,
497                                                   self.y1,pen=pen1)
498         self.timer1 = QtCore.QTimer()
499         self.timer1.setInterval(50)
500         self.timer1.timeout.connect(self.update_plot_data1)
501         self.timer1.start()
502         f_layout.addWidget(self.graphWidget1)
503         vderlayout.addLayout(f_layout)
504         # Gráfica del sensor 2
505         s_layout = QVBoxLayout()
506         self.second = LabelledName('Segundo canal', '0', s_layout)
507         self.graphWidget2 = pg.PlotWidget()
508         self.x2 = list(range(200))  # 200 time points
509         self.y2 = [randint(0,100) for _ in range(200)]
510         self.graphWidget2.setBackground('w')
511         pen2 = pg.mkPen(color=(0, 255, 0))
512         #self.graphWidget2.setGeometry(0, 0, 200, 300)
513         self.data_line2 =  self.graphWidget2.plot(self.x2,
514                                                   self.y2,pen=pen2)
515         self.timer2 = QtCore.QTimer()
516         self.timer2.setInterval(50)
517         self.timer2.timeout.connect(self.update_plot_data2)
518         self.timer2.start()
519         s_layout.addWidget(self.graphWidget2)
520         vderlayout.addLayout(s_layout)
521         # Gráfica del sensor 3
522         t_layout = QVBoxLayout()
523         self.third = LabelledName('Tercer canal', '0', t_layout)
524         self.graphWidget3 = pg.PlotWidget()
525         self.x3 = list(range(200))  # 200 time points
526         self.y3 = [randint(0,100) for _ in range(200)]
527         self.graphWidget3.setBackground('w')
528         pen3 = pg.mkPen(color=(0, 0, 255))
```

```
529         #self.graphWidget3.setGeometry(0, 0, 200, 300)
530         self.data_line3 =  self.graphWidget3.plot(self.x3,
531                                                   self.y3,pen=pen3)
532         self.timer3 = QtCore.QTimer()
533         self.timer3.setInterval(50)
534         self.timer3.timeout.connect(self.update_plot_data3)
535         self.timer3.start()
536         t_layout.addWidget(self.graphWidget3)
537         vderlayout.addLayout(t_layout)
538         # Empezamos el hilo de adquisición
539         self.threadAdq = AdqThread()
540         self.threadAdq.adq_signals.connect(self.update_signal)
541         self.threadAdq.start() # Empezamos el hilo
542         # Empezamos el hilo de predicción
543         self.threadPre = PreThread()
544         self.threadPre.pre_signals.connect(self.predict_signal)
545         self.threadPre.start() # Empezamos el hilo
546         # Se muestra el Box Principal
547         vmainlayout.addStretch()
548         self.show()
549 
550     # Función del hilo de adquisición ------------------------------
551     def closeEvent(self, event):
552         self.threadAdq.stop()
553         event.accept()
554     @pyqtSlot(str,str,str,np.ndarray)
555     def update_signal(self,nombreX,nombreY,nombreZ,matrizActual):
556         # Actualiza las labels y las barras
557         global a
558         global estadoPro, estadoProAnt
559         global acc, loss
560         if estadoPro != estadoProAnt:
561             # Procesamiento
562             if estadoPro == 1:
563                 self.processing.setState("Procesando ...")
564                 self.timer1.setInterval(5000)
565                 self.timer2.setInterval(5000)
566                 self.timer3.setInterval(5000)
567             elif estadoPro == 2:
568                 self.processing.setState("Finalizó procesamiento")
569                 self.timer1.setInterval(50)
570                 self.timer2.setInterval(50)
571                 self.timer3.setInterval(50)
572             elif estadoPro == 3:
573                 self.training.setState("Entrenando ...")
574                 self.timer1.setInterval(5000)
575                 self.timer2.setInterval(5000)
576                 self.timer3.setInterval(5000)
577             elif estadoPro == 4:
578                 self.training.setState("Finalizó entrenamiento")
579                 self.timer1.setInterval(50)
580                 self.timer2.setInterval(50)
581                 self.timer3.setInterval(50)
```

```
582                 self.labelPre.setText("Precisión: " +acc +
583                                       "        "+loss)
584             estadoProAnt = estadoPro
585         # Actualizamos los gráficos
586         try:
587             self.first.setValue(nombreX)
588             self.second.setValue(nombreY)
589             self.third.setValue(str(int(nombreZ)))
590         except:
591             print("error")
592     # Función del hilo de predicción ---------------------------
593     def closeEventP(self, eventp):
594         self.threadPre.stop()
595         eventp.accept()
596     @pyqtSlot(str)
597     def predict_signal(self,Npredict):
598         # Loop de 0.5 s
599         global estadoAdq, estadoAdqAnt
600         global namemove1, namemove2, namemove3, namemove4, namemove5
601         namemove1 = self.move1.getValue()
602         namemove2 = self.move2.getValue()
603         namemove3 = self.move3.getValue()
604         namemove4 = self.move4.getValue()
605         namemove5 = self.move5.getValue()
606         if estadoAdq != estadoAdqAnt:
607             # Movimiento 1, 2, 3, 4 o 5
608             if estadoAdq == 1:
609                 self.move1.setState("Adquiriendo en "+namemove1+".csv")
610             elif estadoAdq == 2:
611                 self.move1.setState("Finalizó la adquisición en "
612                                     +namemove1+".csv")
613             elif estadoAdq == 3:
614                 self.move2.setState("Adquiriendo en "+namemove2+".csv")
615             elif estadoAdq == 4:
616                 self.move2.setState("Finalizó la adquisición en "
617                                     +namemove2+".csv")
618             elif estadoAdq == 5:
619                 self.move3.setState("Adquiriendo en "+namemove3+".csv")
620             elif estadoAdq == 6:
621                 self.move3.setState("Finalizó la adquisición en "
622                                     +namemove3+".csv")
623             elif estadoAdq == 7:
624                 self.move4.setState("Adquiriendo en "+namemove4+".csv")
625             elif estadoAdq == 8:
626                 self.move4.setState("Finalizó la adquisición en "
627                                     +namemove4+".csv")
628             elif estadoAdq == 9:
629                 self.move5.setState("Adquiriendo en "+namemove5+".csv")
630             elif estadoAdq == 10:
631                 self.move5.setState("Finalizó la adquisición en "
632                                     +namemove5+".csv")
633             estadoAdqAnt = estadoAdq
634         self.prediction.setState(Npredict)
```

```
635         # Función de la data actualizada -------------------------
636     def update_plot_data1(self):
637         self.x1 = self.x1[1:]  # remover el primer dato
638         self.x1.append(self.x1[-1] + 1)
639         self.y1 = self.y1[1:]  # remover el primer dato
640         self.y1.append(int(self.first.getValue()))
641         self.data_line1.setData(self.x1, self.y1)  # actualizar
642     def update_plot_data2(self):
643         self.x2 = self.x2[1:]  # remover el primer dato
644         self.x2.append(self.x2[-1] + 1)
645         self.y2 = self.y2[1:]  # remover el primer dato
646         self.y2.append(int(self.second.getValue()))
647         self.data_line2.setData(self.x2, self.y2)  # actualizar
648     def update_plot_data3(self):
649         self.x3 = self.x3[1:]  # remover el primer dato
650         self.x3.append(self.x2[-1] + 1)
651         self.y3 = self.y3[1:]  # remover el primer dato
652         self.y3.append(int(self.third.getValue()))
653         self.data_line3.setData(self.x3, self.y3)  # actualizar
654     # Boton de predecir ---------------------------------------
655     def addButtonPredict(self, parentLayout):
656         self.button = QPushButton("Predecir")
657         self.button.clicked.connect(self.buttonAction)
658         hlayout = QHBoxLayout()
659         hlayout.addWidget(self.button)
660         parentLayout.addLayout(hlayout)
661     def buttonAction(self):
662         pico.write(b"3\n")
663     # Boton para derecha --------------------------------------
664     def addButtonRight(self, parentLayout):
665         self.buttonS = QPushButton("Derecha")
666         self.buttonS.clicked.connect(self.buttonSend)
667         hlayout = QHBoxLayout()
668         hlayout.addWidget(self.buttonS)
669         parentLayout.addLayout(hlayout)
670     def buttonSend(self):
671         pico.write(b"6\n")
672     # Boton Mov1 ----------------------------------------
673     def buttonfM1(self):
674         global FLAG_SAVE_1
675         global a1, Datos, namemove1
676         ## Bandera para guardar
677         FLAG_SAVE_1 = 1
678         # Adquirimos el nombre donde se va guardar
679         namemove1 = self.move1.getValue()
680         print("Empezó la adquisición datos 1")
681         a1 = Datos
682     # Boton Mov2 ------------------------------------------------
683     def buttonfM2(self):
684         global FLAG_SAVE_2
685         global a2, Datos, namemove2
686         ## Bandera para guardar
687         FLAG_SAVE_2 = 1
```

```
688         # Adquirimos el nombre donde se va guardar
689         namemove2 = self.move2.getValue()
690         print("Empezó la adquisición datos 2")
691         a2 = Datos
692     # Boton Mov3 ------------------------------------------------
693     def buttonfM3(self):
694         global FLAG_SAVE_3
695         global a3, Datos, namemove3
696         ## Bandera para guardar
697         FLAG_SAVE_3 = 1
698         # Adquirimos el nombre donde se va guardar
699         namemove3 = self.move3.getValue()
700         print("Empezó la adquisición datos 3")
701         a3 = Datos
702     # Boton Mov4 ------------------------------------------------
703     def buttonfM4(self):
704         global FLAG_SAVE_4
705         global a4, Datos, namemove4
706         ## Bandera para guardar
707         FLAG_SAVE_4 = 1
708         # Adquirimos el nombre donde se va guardar
709         namemove4 = self.move4.getValue()
710         print("Empezó la adquisición datos 4")
711         a4 = Datos
712     # Boton Mov5 ------------------------------------------------
713     def buttonfM5(self):
714         global FLAG_SAVE_5
715         global a5, Datos, namemove5
716         ## Bandera para guardar
717         FLAG_SAVE_5 = 1
718         # Adquirimos el nombre donde se va guardar
719         namemove5 = self.move5.getValue()
720         print("Empezó la adquisición datos 5")
721         a5 = Datos
722     # Boton de procesamiento ------------------------------------
723     def buttonProcess(self):
724         global FLAG_PROCESSING
725         # Bandera para procesar
726         FLAG_PROCESSING = 1
727         self.processing.setState("Procesando ...")
728         print("Empezó el procesamiento")
729     # Boton de entrenamiento ------------------------------------
730     def buttonTrain(self):
731         global FLAG_TRAINING
732         # Bandera para procesar
733         FLAG_TRAINING = 1
734         self.training.setState("Entrenando ...")
735         print("Empezó el entrenamiento")
736     # Boton de predicción ---------------------------------------
737     def buttonPredict(self):
738         global FLAG_PREDICTING, FLAG_TIME
739         global S1, S2, S3, model
740         if FLAG_PREDICTING == 0:
```

```
741                 # Valores de la normalización
742                 S1 = load('S1.bin')
743                 S2 = load('S2.bin')
744                 S3 = load('S3.bin')
745                 model = torch.jit.load('mobilenet_v2model.pt')
746                 self.prediction.setState("Prediciendo")
747                 print("Empezó la predicción")
748             FLAG_PREDICTING = 1
749             FLAG_TIME = FLAG_TIME + 1
750         # Reconectar Pico ------------------------------------------------
751         def addButtonReconnect(self, parentLayout):
752             self.buttonC = QPushButton("Conectar")
753             self.buttonC.clicked.connect(self.buttonConnect)
754             vlayout = QVBoxLayout()
755             vlayout.setContentsMargins(7,5,0,20)
756             vlayout.addWidget(self.buttonC)
757             parentLayout.addLayout(vlayout)
758         def buttonConnect(self):
759             pico = serial.Serial('COM9', 230400)
760             time.sleep(2)
761         # Desconectar Arduino--------------------------------------------
762         def addButtonDisconnect(self, parentLayout):
763             self.buttonD = QPushButton("Desconectar")
764             self.buttonD.clicked.connect(self.buttonDisconnect)
765             vlayout = QVBoxLayout()
766             vlayout.setContentsMargins(0,5,7,20)
767             vlayout.addWidget(self.buttonD)
768             parentLayout.addLayout(vlayout)
769         def buttonDisconnect(self):
770             pico.close()
771 if __name__ == '__main__':
772     # Create the Qt Application
773     app = QApplication(sys.argv)
774     demo = Demo() # <<-- Create an instance
775     demo.show()
        sys.exit(app.exec_())
```

A.4. CONTROLADOR DE LA SIMULACIÓN EN WEBOTS

```
 1 ## código de la simulación - cliente 2-----------------------------
 2 from controller import Robot
 3 import socket
 4 import time
 5 from simple_pid import PID
 6 import sys
 7
 8 # puerto de conexión
 9 host = '127.0.0.1'
10 port = 1233
11
12 ClientSocket = socket.socket()
13 print('Waiting for connection')
```

```
try:
    ClientSocket.connect((host, port))
except socket.error as e:
    print(str(e))
Response = ClientSocket.recv(1024)
i = 0
# create the Robot instance.
robot = Robot()
timestep = 64
max_speed = 6.28

# index finger motors
m_MCP_index = robot.getMotor('motor_MCP_index')
m_PIP_index = robot.getMotor('motor_PIP_index')
m_DIP_index = robot.getMotor('motor_DIP_index')
# middle finger motors
m_MCP_middle = robot.getMotor('motor_MCP_middle')
m_PIP_middle = robot.getMotor('motor_PIP_middle')
m_DIP_middle = robot.getMotor('motor_DIP_middle')
# ring finger motors
m_MCP_ring = robot.getMotor('motor_MCP_ring')
m_PIP_ring = robot.getMotor('motor_PIP_ring')
m_DIP_ring = robot.getMotor('motor_DIP_ring')
# Little finger motors
m_MCP_little = robot.getMotor('motor_MCP_little')
m_PIP_little = robot.getMotor('motor_PIP_little')
m_DIP_little = robot.getMotor('motor_DIP_little')
# thumb motors
m_MCP_1_thumb = robot.getMotor('motor_MCP_1_thumb')
m_X_thumb = robot.getMotor('motor_X_thumb')
m_MCP_2_thumb = robot.getMotor('motor_MCP_2_thumb')
m_MCP_3_thumb = robot.getMotor('motor_MCP_3_thumb')
m_IP_thumb = robot.getMotor('motor_IP_thumb')
m_S1_thumb = robot.getMotor('motor_S1_thumb')

# Start position and velocity of the index finger motors
m_MCP_index.setPosition(float('inf'))
m_MCP_index.setVelocity(0.0)
m_PIP_index.setPosition(float('inf'))
m_PIP_index.setVelocity(0.0)
m_DIP_index.setPosition(float('inf'))
m_DIP_index.setVelocity(0.0)
# Start position and velocity of the middle finger motors
m_MCP_middle.setPosition(float('inf'))
m_MCP_middle.setVelocity(0.0)
m_PIP_middle.setPosition(float('inf'))
m_PIP_middle.setVelocity(0.0)
m_DIP_middle.setPosition(float('inf'))
m_DIP_middle.setVelocity(0.0)
# Start position and velocity of the ring finger motors
m_MCP_ring.setPosition(float('inf'))
m_MCP_ring.setVelocity(0.0)
m_PIP_ring.setPosition(float('inf'))
```

```
67 m_PIP_ring.setVelocity(0.0)
68 m_DIP_ring.setPosition(float('inf'))
69 m_DIP_ring.setVelocity(0.0)
70 # Start position and velocity of the little finger motors
71 m_MCP_little.setPosition(float('inf'))
72 m_MCP_little.setVelocity(0.0)
73 m_PIP_little.setPosition(float('inf'))
74 m_PIP_little.setVelocity(0.0)
75 m_DIP_little.setPosition(float('inf'))
76 m_DIP_little.setVelocity(0.0)
77 # Start position and velocity of the thumb motors
78 m_MCP_1_thumb.setPosition(float('inf'))
79 m_MCP_1_thumb.setVelocity(0.0)
80 m_X_thumb.setPosition(float('inf'))
81 m_X_thumb.setVelocity(0.0)
82 m_MCP_2_thumb.setPosition(float('inf'))
83 m_MCP_2_thumb.setVelocity(0.0)
84 m_MCP_3_thumb.setPosition(float('inf'))
85 m_MCP_3_thumb.setVelocity(0.0)
86 m_IP_thumb.setPosition(float('inf'))
87 m_IP_thumb.setVelocity(0.0)
88 m_S1_thumb.setPosition(float('inf'))
89 m_S1_thumb.setVelocity(0.0)
90
91 # Position sensors of the index finger
92 s_MCP_index = robot.getPositionSensor('pos_MCP_index')
93 s_MCP_index.enable(timestep)
94 s_PIP_index = robot.getPositionSensor('pos_PIP_index')
95 s_PIP_index.enable(timestep)
96 s_DIP_index = robot.getPositionSensor('pos_DIP_index')
97 s_DIP_index.enable(timestep)
98 # Position sensors of the middle finger
99 s_MCP_middle = robot.getPositionSensor('pos_MCP_middle')
100 s_MCP_middle.enable(timestep)
101 s_PIP_middle = robot.getPositionSensor('pos_PIP_middle')
102 s_PIP_middle.enable(timestep)
103 s_DIP_middle = robot.getPositionSensor('pos_DIP_middle')
104 s_DIP_middle.enable(timestep)
105 # Position sensors of the ring finger
106 s_MCP_ring = robot.getPositionSensor('pos_MCP_ring')
107 s_MCP_ring.enable(timestep)
108 s_PIP_ring = robot.getPositionSensor('pos_PIP_ring')
109 s_PIP_ring.enable(timestep)
110 s_DIP_ring = robot.getPositionSensor('pos_DIP_ring')
111 s_DIP_ring.enable(timestep)
112 # Position sensors of the little finger
113 s_MCP_little = robot.getPositionSensor('pos_MCP_little')
114 s_MCP_little.enable(timestep)
115 s_PIP_little = robot.getPositionSensor('pos_PIP_little')
116 s_PIP_little.enable(timestep)
117 s_DIP_little = robot.getPositionSensor('pos_DIP_little')
118 s_DIP_little.enable(timestep)
119 # Position sensors of the thumb
```

```
120 s_MCP_1_thumb = robot.getPositionSensor('pos_MCP_1_thumb')
121 s_MCP_1_thumb.enable(timestep)
122 s_X_thumb = robot.getPositionSensor('pos_X_thumb')
123 s_X_thumb.enable(timestep)
124 s_MCP_2_thumb = robot.getPositionSensor('pos_MCP_2_thumb')
125 s_MCP_2_thumb.enable(timestep)
126 s_MCP_3_thumb = robot.getPositionSensor('pos_MCP_3_thumb')
127 s_MCP_3_thumb.enable(timestep)
128 s_IP_thumb = robot.getPositionSensor('pos_IP_thumb')
129 s_IP_thumb.enable(timestep)
130 s_S1_thumb = robot.getPositionSensor('pos_S1_thumb')
131 s_S1_thumb.enable(timestep)
132
133 # index finger PID
134 pid_MCP_index = PID(4, 10**-12, 0, setpoint = 0)
135 pid_PIP_index = PID(4, 10**-12, 0, setpoint = 0)
136 pid_DIP_index = PID(4, 10**-12, 0, setpoint = 0)
137 # middle finger PID
138 pid_MCP_middle = PID(4, 10**-12, 0, setpoint = 0)
139 pid_PIP_middle = PID(4, 10**-12, 0, setpoint = 0)
140 pid_DIP_middle = PID(4, 10**-12, 0, setpoint = 0)
141 # ring finger PID
142 pid_MCP_ring = PID(4, 10**-12, 0, setpoint = 0)
143 pid_PIP_ring = PID(4, 10**-12, 0, setpoint = 0)
144 pid_DIP_ring = PID(4, 10**-12, 0, setpoint = 0)
145 # little finger PID
146 pid_MCP_little = PID(4, 10**-12, 0, setpoint = 0)
147 pid_PIP_little = PID(4, 10**-12, 0, setpoint = 0)
148 pid_DIP_little = PID(4, 10**-12, 0, setpoint = 0)
149 # thumb PID
150 pid_MCP_1_thumb = PID(4, 10**-12, 0, setpoint = 0)
151 pid_X_thumb = PID(4, 10**-12, 0, setpoint = 0)
152 pid_MCP_2_thumb = PID(4, 10**-12, 0, setpoint = 0)
153 pid_MCP_3_thumb = PID(4, 10**-12, 0, setpoint = 0)
154 pid_IP_thumb = PID(4, 10**-12, 0, setpoint = 0)
155 pid_S1_thumb = PID(4, 10**-12, 0, setpoint = 0)
156
157 speed = 1
158 label_ant = 0.0
159 k = 0 # to get position sensor reading
160
161 while robot.step(timestep) != -1:
162
163     millisec = time.time()*1000
164
165     # PID of the motors of the index finger
166     m_MCP_index.setVelocity(pid_MCP_index(s_MCP_index.getValue()))
167     m_PIP_index.setVelocity(pid_PIP_index(s_PIP_index.getValue()))
168     m_DIP_index.setVelocity(pid_DIP_index(s_DIP_index.getValue()))
169     # PID of the motors of the middle finger
170     m_MCP_middle.setVelocity(pid_MCP_middle(s_MCP_middle.getValue()))
171     m_PIP_middle.setVelocity(pid_PIP_middle(s_PIP_middle.getValue()))
172     m_DIP_middle.setVelocity(pid_DIP_middle(s_DIP_middle.getValue()))
```

```
173     # PID of the motors of the ring finger
174     m_MCP_ring.setVelocity(pid_MCP_ring(s_MCP_ring.getValue()))
175     m_PIP_ring.setVelocity(pid_PIP_ring(s_PIP_ring.getValue()))
176     m_DIP_ring.setVelocity(pid_DIP_ring(s_DIP_ring.getValue()))
177     # PID of the motors of the little finger
178     m_MCP_little.setVelocity(pid_MCP_little(s_MCP_little.getValue()))
179     m_PIP_little.setVelocity(pid_PIP_little(s_PIP_little.getValue()))
180     m_DIP_little.setVelocity(pid_DIP_little(s_DIP_little.getValue()))
181     # PID of the motors of the thumb
182     m_MCP_1_thumb.setVelocity(pid_MCP_1_thumb(s_MCP_1_thumb.getValue()))
183     m_X_thumb.setVelocity(pid_X_thumb(s_X_thumb.getValue()))
184     m_MCP_2_thumb.setVelocity(pid_MCP_2_thumb(s_MCP_2_thumb.getValue()))
185     m_MCP_3_thumb.setVelocity(pid_MCP_3_thumb(s_MCP_3_thumb.getValue()))
186     m_IP_thumb.setVelocity(pid_IP_thumb(s_IP_thumb.getValue()))
187     m_S1_thumb.setVelocity(pid_S1_thumb(s_S1_thumb.getValue()))
188
189     # Receive the label
190     i = i + 1
191     ClientSocket.send(str.encode(str(i)))
192     Response = ClientSocket.recv(1024)
193     label = float(Response)
194
195     # Set point label
196     if (label_ant != label):
197
198         if (label == 0.0):
199             # Mano relajada
200             pid_MCP_index.setpoint = (10 - 1.5)*3.1416/180
201             pid_PIP_index.setpoint = (23.76 - 14.94)*3.1416/180
202             pid_DIP_index.setpoint = (29.84 - 13.14)*3.1416/180
203             pid_MCP_middle.setpoint = (10 - 2.25)*3.1416/180
204             pid_PIP_middle.setpoint = (23.76 - 15.73)*3.1416/180
205             pid_DIP_middle.setpoint = (29.84 - 14.88)*3.1416/180
206             pid_MCP_ring.setpoint = (10 - 0.45)*3.1416/180
207             pid_PIP_ring.setpoint = (23.76 - 13.83)*3.1416/180
208             pid_DIP_ring.setpoint = (29.84 - 10.59)*3.1416/180
209             pid_MCP_little.setpoint = (10 + 1)*3.1416/180
210             pid_PIP_little.setpoint = (23.76 - 12.29)*3.1416/180
211             pid_DIP_little.setpoint = (29.84 - 6.75)*3.1416/180
212             pid_MCP_1_thumb.setpoint = (15)*3.1416/180
213             pid_X_thumb.setpoint = 2
214             pid_MCP_2_thumb.setpoint = (119.62 - 133.3)*3.1416/180
215             pid_MCP_3_thumb.setpoint = (116.99 - 117.84)*3.1416/180
216             pid_IP_thumb.setpoint = (64.61 - 69.43)*3.1416/180
217             pid_S1_thumb.setpoint = (61.48 - 74.86)*3.1416/180
218
219         elif (label == 1.0):
220             # Empuje de plataforma
221             pid_MCP_index.setpoint = (-1 - 1.5)*3.1416/180
222             pid_PIP_index.setpoint = (12.29 - 14.94)*3.1416/180
223             pid_DIP_index.setpoint = (6.75 - 13.14)*3.1416/180
224             pid_MCP_middle.setpoint = (-1 - 2.25)*3.1416/180
225             pid_PIP_middle.setpoint = (12.29 - 15.73)*3.1416/180
```

```
226             pid_DIP_middle.setpoint = (6.75 - 14.88)*3.1416/180
227             pid_MCP_ring.setpoint = (-1 - 0.45)*3.1416/180
228             pid_PIP_ring.setpoint = (12.29 - 13.83)*3.1416/180
229             pid_DIP_ring.setpoint = (6.75 - 10.59)*3.1416/180
230             pid_MCP_little.setpoint = (-1 + 1)*3.1416/180
231             pid_PIP_little.setpoint = (12.29 - 12.29)*3.1416/180
232             pid_DIP_little.setpoint = (6.75 - 6.75)*3.1416/180
233             pid_MCP_1_thumb.setpoint = (0)*3.1416/180
234             pid_X_thumb.setpoint = 0
235             pid_MCP_2_thumb.setpoint = (133.3 - 133.3)*3.1416/180
236             pid_MCP_3_thumb.setpoint = (117.84 - 117.84)*3.1416/180
237             pid_IP_thumb.setpoint = (69.43 - 69.43)*3.1416/180
238             pid_S1_thumb.setpoint = (74.86 - 74.86)*3.1416/180
239
240         elif (label == 2.0):
241             # Pulgar aducido
242             pid_MCP_index.setpoint = (72 - 1.5)*3.1416/180
243             pid_PIP_index.setpoint = (77.7 - 14.94)*3.1416/180
244             pid_DIP_index.setpoint = (87.24 - 13.14)*3.1416/180
245             pid_MCP_middle.setpoint = (72 - 2.25)*3.1416/180
246             pid_PIP_middle.setpoint = (77.7 - 15.73)*3.1416/180
247             pid_DIP_middle.setpoint = (87.24 - 14.88)*3.1416/180
248             pid_MCP_ring.setpoint = (72 - 0.45)*3.1416/180
249             pid_PIP_ring.setpoint = (77.7 - 13.83)*3.1416/180
250             pid_DIP_ring.setpoint = (87.24 - 10.59)*3.1416/180
251             pid_MCP_little.setpoint = (72 + 1)*3.1416/180
252             pid_PIP_little.setpoint = (77.7 - 12.29)*3.1416/180
253             pid_DIP_little.setpoint = (87.24 - 6.75)*3.1416/180
254             pid_MCP_1_thumb.setpoint = (0)*3.1416/180
255             pid_X_thumb.setpoint = 0
256             pid_MCP_2_thumb.setpoint = (133.3 - 133.3)*3.1416/180
257             pid_MCP_3_thumb.setpoint = (117.84 - 117.84)*3.1416/180
258             pid_IP_thumb.setpoint = (69.43 - 69.43)*3.1416/180
259             pid_S1_thumb.setpoint = (74.86 - 74.86)*3.1416/180
260
261         elif (label == 3.0):
262             # Pinza lateral
263             pid_MCP_index.setpoint = (80.65 - 1.5)*3.1416/180
264             pid_PIP_index.setpoint = (83.74 - 14.94)*3.1416/180
265             pid_DIP_index.setpoint = (91.22 - 13.14)*3.1416/180
266             pid_MCP_middle.setpoint = (80.65 - 2.25)*3.1416/180
267             pid_PIP_middle.setpoint = (83.74 - 15.73)*3.1416/180
268             pid_DIP_middle.setpoint = (91.22 - 14.88)*3.1416/180
269             pid_MCP_ring.setpoint = (80.65 - 0.45)*3.1416/180
270             pid_PIP_ring.setpoint = (83.74 - 13.83)*3.1416/180
271             pid_DIP_ring.setpoint = (91.22 - 10.59)*3.1416/180
272             pid_MCP_little.setpoint = (80.65 + 1)*3.1416/180
273             pid_PIP_little.setpoint = (83.74 - 12.29)*3.1416/180
274             pid_DIP_little.setpoint = (91.22 - 6.75)*3.1416/180
275             pid_MCP_1_thumb.setpoint = (0)*3.1416/180
276             pid_X_thumb.setpoint = 6.5
277             pid_MCP_2_thumb.setpoint = (74.74 - 133.3)*3.1416/180
278             pid_MCP_3_thumb.setpoint = (121.73 - 117.84)*3.1416/180
```

```
279             pid_IP_thumb.setpoint = (50 - 69.43)*3.1416/180
280             pid_S1_thumb.setpoint = (22.03 - 74.86)*3.1416/180
281 
282         elif (label == 4.0):
283             # Pinza pulgar-índice
284             pid_MCP_index.setpoint = (40 - 1.5)*3.1416/180
285             pid_PIP_index.setpoint = (52.19 - 14.94)*3.1416/180
286             pid_DIP_index.setpoint = (65.69 - 13.14)*3.1416/180
287             pid_MCP_middle.setpoint = (15 - 2.25)*3.1416/180
288             pid_PIP_middle.setpoint = (28.8 - 15.73)*3.1416/180
289             pid_DIP_middle.setpoint = (37.63 - 14.88)*3.1416/180
290             pid_MCP_ring.setpoint = (15 - 0.45)*3.1416/180
291             pid_PIP_ring.setpoint = (28.8 - 13.83)*3.1416/180
292             pid_DIP_ring.setpoint = (37.63 - 10.59)*3.1416/180
293             pid_MCP_little.setpoint = (15 + 1)*3.1416/180
294             pid_PIP_little.setpoint = (28.8 - 12.29)*3.1416/180
295             pid_DIP_little.setpoint = (37.63 - 6.75)*3.1416/180
296             pid_MCP_1_thumb.setpoint = (84.5)*3.1416/180
297             pid_X_thumb.setpoint = 4.3
298             pid_MCP_2_thumb.setpoint = (97.89 - 133.3)*3.1416/180
299             pid_MCP_3_thumb.setpoint = (117.97- 117.84)*3.1416/180
300             pid_IP_thumb.setpoint = (57.96 - 69.43)*3.1416/180
301             pid_S1_thumb.setpoint = (42.47 - 74.86)*3.1416/180
302 
303         label_ant = label
304     pass
305     millisec_new = time.time()*1000
306     print(millisec_new - millisec)
```

A.5. SERVIDOR MULTIHILO

```
1  # SERVIDOR MULTIHILO
2  # Importar librerías
3  import socket
4  from _thread import *
5  # Red local
6  host = '127.0.0.1'
7  port = 1233
8  ThreadCount = 0
9  numberr = '0'
10 # Conexión del servidor con la GUI
11 def client_handler_1(connection):
12     global numberr
13     connection.send(str.encode('conección con el servidor'))
14     while True:
15         data = connection.recv(2048)
16         numberr = data.decode('utf-8')
17         #print(numberr)
18         if numberr == 'BYE':
19             break
20         reply = f'number of the client 1: {numberr}'
21         connection.sendall(str.encode(reply))
```

```
22      connection.close()
23  # Conexión del servidor con la simulación
24  def client_handler_2(connection):
25      global numberr
26      connection.send(str.encode('conección con el servidor'))
27      while True:
28          data = connection.recv(2048)
29          message = data.decode('utf-8')
30          print(message)
31          if message == 'BYE':
32              break
33          connection.sendall(str.encode(numberr))
34      connection.close()
35  # Empezar le servidor
36  ServerSocket = socket.socket()
37  try:
38      ServerSocket.bind((host, port))
39  except socket.error as e:
40      print(str(e))
41  print(f'Server is listing on the port {port}...')
42  ServerSocket.listen()
43  num_client = 1 # la GUI
44  while True:
45      if num_client == 1:
46          Client_1, address_1 = ServerSocket.accept()
47          print('Connected to: ' + address_1[0] +
48                ':' + str(address_1[1]) + ' number of client: '
49                + str(num_client))
50          start_new_thread(client_handler_1, (Client_1, ))
51      elif num_client == 2:
52          Client_2, address_2 = ServerSocket.accept()
53          print('Connected to: ' + address_2[0] +
54                ':' + str(address_2[1]) + ' number of client: '
55                + str(num_client))
56          start_new_thread(client_handler_2, (Client_2, ))
57      num_client = num_client + 1 # la simulación
```

ANEXO B

CONSENTIMIENTOS INFORMADOS

Consentimiento Informado para Participantes de Investigación de Tesis

El propósito de esta ficha de consentimiento es proveer a los participantes en esta investigación con una clara explicación de la naturaleza de la misma, así como de su rol en ella como participantes.

La presente investigación es conducida por Edgard Jesús Barazorda Rodríguez, de la Universidad Nacional de Ingeniería.

Si usted accede a participar en este estudio, se le pedirá ser parte de los experimentos de la investigación con relación a las señales mioléctricas y responder las preguntas indicadas en este documento. Esto tomará aproximadamente 25 minutos de su tiempo. La sesión se grabará, de modo que el investigador pueda reunir la información en la experimentación.

La participación es este estudio es estrictamente voluntaria. La información que se recoja no se usará para ningún otro propósito fuera de los de esta investigación.

Si tiene alguna duda sobre este proyecto, puede hacer preguntas en cualquier momento durante su participación en él. Igualmente, puede retirarse del proyecto en cualquier momento sin que eso lo perjudique en ninguna forma. Si alguno de los experimentos le parece incómodo, tiene usted el derecho de hacérselo saber al investigador o no realizarlos.

Desde ya le agradecemos su participación.

Acepto participar voluntariamente en esta investigación, conducida por Edgard Jesús Barazorda Rodríguez y responder las preguntas indicadas en este documento.

Me han indicado también que tendré que realizar experimentos con relación a las señales mioléctricas, lo cual tomará aproximadamente 25 minutos.

Reconozco que la información que yo provea en el curso de esta investigación no será usada para ningún otro propósito fuera de los de este estudio sin mi consentimiento. He sido informado de que puedo hacer preguntas sobre el proyecto en cualquier momento y que puedo retirarme del mismo cuando así lo decida, sin que esto acarree perjuicio alguno para mi persona.

Entiendo que una copia de esta ficha de consentimiento me será entregada, y que puedo pedir información sobre los resultados de este estudio cuando éste haya concluido. Para esto, puedo contactar al celular 949785665.

Datos del participante:

Nombres y apellidos: _____

Edad: _____ Sexo:_____

Preguntas:

¿Posee alguna enfermedad muscular? _____. Si es sí, especifique cuál: _____

¿Sintió alguna molestia durante los experimentos realizado? _____. Si es sí, especifique cuál:

_____

¿Qué sugeriría para mejorar el procedimiento experimental?

_____

---------- ----------

Firma del Participante Fecha

Consentimiento Informado para Participantes de Investigación de Tesis

El propósito de esta ficha de consentimiento es proveer a los participantes en esta investigación con una clara explicación de la naturaleza de la misma, así como de su rol en ella como participantes.

La presente investigación es conducida por Edgard Jesús Barazorda Rodríguez, de la Universidad Nacional de Ingeniería.

Si usted accede a participar en este estudio, se le pedirá ser parte de los experimentos de la investigación con relación a las señales mioléctricas y responder las preguntas indicadas en este documento. Esto tomará aproximadamente 25 minutos de su tiempo. La sesión se grabará, de modo que el investigador pueda reunir la información en la experimentación.

La participación es este estudio es estrictamente voluntaria. La información que se recoja no se usará para ningún otro propósito fuera de los de esta investigación.

Si tiene alguna duda sobre este proyecto, puede hacer preguntas en cualquier momento durante su participación en él. Igualmente, puede retirarse del proyecto en cualquier momento sin que eso lo perjudique en ninguna forma. Si alguno de los experimentos le parece incómodo, tiene usted el derecho de hacérselo saber al investigador o no realizarlos.

Desde ya le agradecemos su participación.

Acepto participar voluntariamente en esta investigación, conducida por Edgard Jesús Barazorda Rodríguez y responder las preguntas indicadas en este documento.

Me han indicado también que tendré que realizar experimentos con relación a las señales mioléctricas, lo cual tomará aproximadamente 25 minutos.

Reconozco que la información que yo provea en el curso de esta investigación no será usada para ningún otro propósito fuera de los de este estudio sin mi consentimiento. He sido informado de que puedo hacer preguntas sobre el proyecto en cualquier momento y que puedo retirarme del mismo cuando así lo decida, sin que esto acarree perjuicio alguno para mi persona.

Entiendo que una copia de esta ficha de consentimiento me será entregada, y que puedo pedir información sobre los resultados de este estudio cuando éste haya concluido. Para esto, puedo contactar al celular 949785665.

Datos del participante:

Nombres y apellidos: Jesús Gabriel Julcarima Fuerte

Edad: 25 Sexo: M

Preguntas:

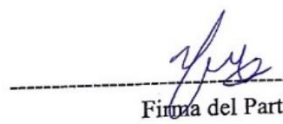
¿Posee alguna enfermedad muscular? NO. Si es sí, especifique cuál: _____

¿Sintió alguna molestia durante los experimentos realizado? NO. Si es sí, especifique cuál:

_____

¿Qué sugeriría para mejorar el procedimiento experimental?

Un mejor soporte para evitar fatiga del brazo



Firma del Participante

30/03/2023

Fecha

Consentimiento Informado para Participantes de Investigación de Tesis

El propósito de esta ficha de consentimiento es proveer a los participantes en esta investigación con una clara explicación de la naturaleza de la misma, así como de su rol en ella como participantes.

La presente investigación es conducida por Edgard Jesús Barazorda Rodríguez, de la Universidad Nacional de Ingeniería.

Si usted accede a participar en este estudio, se le pedirá ser parte de los experimentos de la investigación con relación a las señales mioléctricas y responder las preguntas indicadas en este documento. Esto tomará aproximadamente 25 minutos de su tiempo. La sesión se grabará, de modo que el investigador pueda reunir la información en la experimentación.

La participación es este estudio es estrictamente voluntaria. La información que se recoja no se usará para ningún otro propósito fuera de los de esta investigación.

Si tiene alguna duda sobre este proyecto, puede hacer preguntas en cualquier momento durante su participación en él. Igualmente, puede retirarse del proyecto en cualquier momento sin que eso lo perjudique en ninguna forma. Si alguno de los experimentos le parece incómodo, tiene usted el derecho de hacérselo saber al investigador o no realizarlos.

Desde ya le agradecemos su participación.

Acepto participar voluntariamente en esta investigación, conducida por Edgard Jesús Barazorda Rodríguez y responder las preguntas indicadas en este documento.

Me han indicado también que tendré que realizar experimentos con relación a las señales mioléctricas, lo cual tomará aproximadamente 25 minutos.

Reconozco que la información que yo provea en el curso de esta investigación no será usada para ningún otro propósito fuera de los de este estudio sin mi consentimiento. He sido informado de que puedo hacer preguntas sobre el proyecto en cualquier momento y que puedo retirarme del mismo cuando así lo decida, sin que esto acarree perjuicio alguno para mi persona.

Entiendo que una copia de esta ficha de consentimiento me será entregada, y que puedo pedir información sobre los resultados de este estudio cuando éste haya concluido. Para esto, puedo contactar al celular 949785665.

Datos del participante:

Nombres y apellidos: Andrea Alejandra Palacios Silva

Edad: 21 años Sexo: Femenino

Preguntas:

¿Posee alguna enfermedad muscular? NO. Si es sí, especifique cuál: _____

¿Sintió alguna molestia durante los experimentos realizado? SÍ. Si es sí, especifique cuál:
Un poco de calambre al final de la prueba

¿Qué sugeriría para mejorar el procedimiento experimental?
Mejorar el soporte donde se encuentra apoyado el antebrazo.

Firma del Participante

30 de Marzo del 2023
Fecha

Consentimiento Informado para Participantes de Investigación de Tesis

El propósito de esta ficha de consentimiento es proveer a los participantes en esta investigación con una clara explicación de la naturaleza de la misma, así como de su rol en ella como participantes.

La presente investigación es conducida por Edgard Jesús Barazorda Rodríguez, de la Universidad Nacional de Ingeniería.

Si usted accede a participar en este estudio, se le pedirá ser parte de los experimentos de la investigación con relación a las señales mioléctricas y responder las preguntas indicadas en este documento. Esto tomará aproximadamente 25 minutos de su tiempo. La sesión se grabará, de modo que el investigador pueda reunir la información en la experimentación.

La participación es este estudio es estrictamente voluntaria. La información que se recoja no se usará para ningún otro propósito fuera de los de esta investigación.

Si tiene alguna duda sobre este proyecto, puede hacer preguntas en cualquier momento durante su participación en él. Igualmente, puede retirarse del proyecto en cualquier momento sin que eso lo perjudique en ninguna forma. Si alguno de los experimentos le parece incómodo, tiene usted el derecho de hacérselo saber al investigador o no realizarlos.

Desde ya le agradecemos su participación.

Acepto participar voluntariamente en esta investigación, conducida por Edgard Jesús Barazorda Rodríguez y responder las preguntas indicadas en este documento.

Me han indicado también que tendré que realizar experimentos con relación a las señales mioléctricas, lo cual tomará aproximadamente 25 minutos.

Reconozco que la información que yo provea en el curso de esta investigación no será usada para ningún otro propósito fuera de los de este estudio sin mi consentimiento. He sido informado de que puedo hacer preguntas sobre el proyecto en cualquier momento y que puedo retirarme del mismo cuando así lo decida, sin que esto acarree perjuicio alguno para mi persona.

Entiendo que una copia de esta ficha de consentimiento me será entregada, y que puedo pedir información sobre los resultados de este estudio cuando éste haya concluido. Para esto, puedo contactar al celular 949785665.

Datos del participante:

Nombres y apellidos: Pold Arampa Saravia

Edad: 24 Sexo: Masculino

Preguntas:

¿Posee alguna enfermedad muscular? No. Si es sí, especifique cuál: _____

¿Sintió alguna molestia durante los experimentos realizado? Si. Si es sí, especifique cuál:
Presion, adormecimiento

¿Qué sugeriría para mejorar el procedimiento experimental?
Correa mas suave, que no apriete mucho

Firma del Participante

30-marzo-2023
Fecha

Consentimiento Informado para Participantes de Investigación de Tesis

El propósito de esta ficha de consentimiento es proveer a los participantes en esta investigación con una clara explicación de la naturaleza de la misma, así como de su rol en ella como participantes.

La presente investigación es conducida por Edgard Jesús Barazorda Rodríguez, de la Universidad Nacional de Ingeniería.

Si usted accede a participar en este estudio, se le pedirá ser parte de los experimentos de la investigación con relación a las señales mioléctricas y responder las preguntas indicadas en este documento. Esto tomará aproximadamente 25 minutos de su tiempo. La sesión se grabará, de modo que el investigador pueda reunir la información en la experimentación.

La participación es este estudio es estrictamente voluntaria. La información que se recoja no se usará para ningún otro propósito fuera de los de esta investigación.

Si tiene alguna duda sobre este proyecto, puede hacer preguntas en cualquier momento durante su participación en él. Igualmente, puede retirarse del proyecto en cualquier momento sin que eso lo perjudique en ninguna forma. Si alguno de los experimentos le parece incómodo, tiene usted el derecho de hacérselo saber al investigador o no realizarlos.

Desde ya le agradecemos su participación.

Acepto participar voluntariamente en esta investigación, conducida por Edgard Jesús Barazorda Rodríguez y responder las preguntas indicadas en este documento.

Me han indicado también que tendré que realizar experimentos con relación a las señales mioléctricas, lo cual tomará aproximadamente 25 minutos.

Reconozco que la información que yo provea en el curso de esta investigación no será usada para ningún otro propósito fuera de los de este estudio sin mi consentimiento. He sido informado de que puedo hacer preguntas sobre el proyecto en cualquier momento y que puedo retirarme del mismo cuando así lo decida, sin que esto acarree perjuicio alguno para mi persona.

Entiendo que una copia de esta ficha de consentimiento me será entregada, y que puedo pedir información sobre los resultados de este estudio cuando éste haya concluido. Para esto, puedo contactar al celular 949785665.

Datos del participante:

Nombres y apellidos: Miguel Angel Vera Chavez

Edad: 21 Sexo: M

Preguntas:

¿Posee alguna enfermedad muscular? X. Si es sí, especifique cuál: —

¿Sintió alguna molestia durante los experimentos realizado? Sí. Si es sí, especifique cuál:

Presión

¿Qué sugeriría para mejorar el procedimiento experimental?

Firma del Participante

30/03/2023

Fecha

Consentimiento Informado para Participantes de Investigación de Tesis

El propósito de esta ficha de consentimiento es proveer a los participantes en esta investigación con una clara explicación de la naturaleza de la misma, así como de su rol en ella como participantes.

La presente investigación es conducida por Edgard Jesús Barazorda Rodríguez, de la Universidad Nacional de Ingeniería.

Si usted accede a participar en este estudio, se le pedirá ser parte de los experimentos de la investigación con relación a las señales mioléctricas y responder las preguntas indicadas en este documento. Esto tomará aproximadamente 25 minutos de su tiempo. La sesión se grabará, de modo que el investigador pueda reunir la información en la experimentación.

La participación es este estudio es estrictamente voluntaria. La información que se recoja no se usará para ningún otro propósito fuera de los de esta investigación.

Si tiene alguna duda sobre este proyecto, puede hacer preguntas en cualquier momento durante su participación en él. Igualmente, puede retirarse del proyecto en cualquier momento sin que eso lo perjudique en ninguna forma. Si alguno de los experimentos le parece incómodo, tiene usted el derecho de hacérselo saber al investigador o no realizarlos.

Desde ya le agradecemos su participación.

Acepto participar voluntariamente en esta investigación, conducida por Edgard Jesús Barazorda Rodríguez y responder las preguntas indicadas en este documento.

Me han indicado también que tendré que realizar experimentos con relación a las señales mioléctricas, lo cual tomará aproximadamente 25 minutos.

Reconozco que la información que yo provea en el curso de esta investigación no será usada para ningún otro propósito fuera de los de este estudio sin mi consentimiento. He sido informado de que puedo hacer preguntas sobre el proyecto en cualquier momento y que puedo retirarme del mismo cuando así lo decida, sin que esto acarree perjuicio alguno para mi persona.

Entiendo que una copia de esta ficha de consentimiento me será entregada, y que puedo pedir información sobre los resultados de este estudio cuando éste haya concluido. Para esto, puedo contactar al celular 949785665.

Datos del participante:

Nombres y apellidos: Iván Alejandro Peña Vilca

Edad: 29 Sexo: Masculino

Preguntas:

¿Posee alguna enfermedad muscular? No. Si es sí, especifique cuál: _____

¿Sintió alguna molestia durante los experimentos realizado? No. Si es sí, especifique cuál:

_____

¿Qué sugeriría para mejorar el procedimiento experimental?

Adecuar el posicionamiento inicial del brazo.

Firma del Participante

30 - 03 - 23

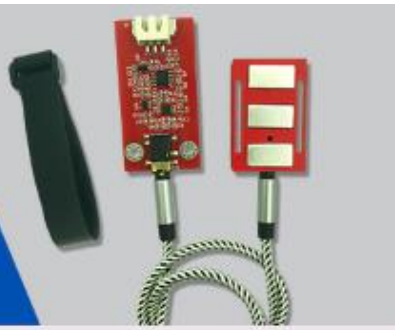
Fecha

ANEXO C

DATOS DE FABRICANTE

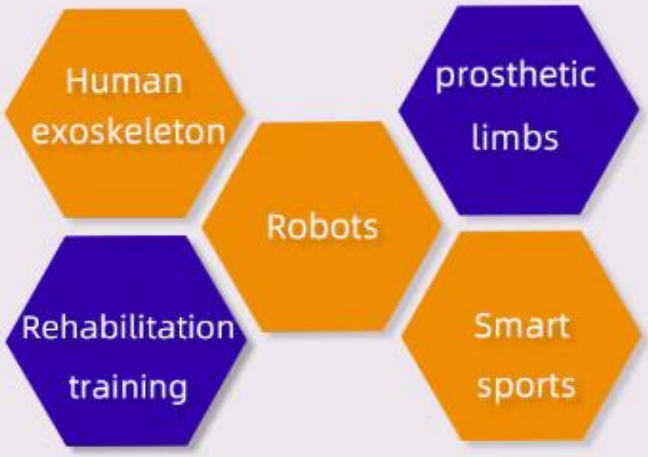
C.1. DRY ELECTRODE SENSOR EMG SICHIRAY

dry electrode electromyography sensor



The sensor integrates the filtering and amplification circuit, amplifies the weak human surface muscle electrical signal within a range of 1.5mV, and effectively suppresses the noise (especially the frequency interference) by differential input and analog filter circuit. The output signal is in the form of analog, with 1.5V as the reference voltage and 0 to 3.0V of the output. The size of the output signal depends on the amount of activity of the selected muscle, the waveform of the output signal can significantly indicate the condition of the subcutaneous muscle situator in the observed position, facilitate the analysis and research of myoelectrical signal, detect muscle activity, such as muscle tightness, strength, fatigue, etc.

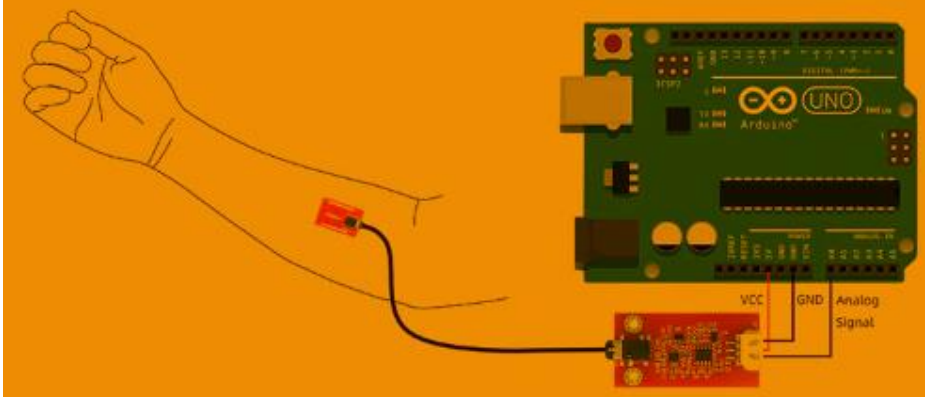
The dry electrode electromyography sensor is an active sensor that provides high-quality signal gathering and is easy to use. Whether used in static or dynamic applications, it takes only some extremely simple preparation. The use of dry electrode guide, no conductive gel can also get good signal quality, so with long life, simple and convenient to use and so on, more suitable for ordinary users, and the use of gel probe medical electrode is usually disposable, more troublesome to use.



demonstration

Demonstrate how to identify the generation of electromyocardial signals by counting the number of grips.

Provide | **Arduino Code** | **description**



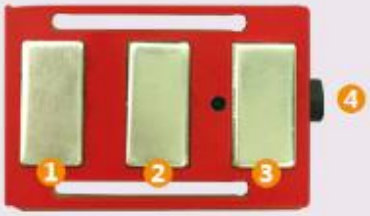
How to wear

Q1: Where can I place dry electrodes on my arm? Are there any requirements?

A1:With a matching three-metal dry electrode plate, you don't need to pay attention to the reference level, just keep the electrode plate in the same direction as the muscles.

Interface Definition

Number	Pin definition
1	Power input negative
2	Power input positive (3.3 ~ 5.5V)
3	Analog signal output (0 ~ 3.0V)
4	PJ-342 dry electrode interface



Dry electrode guide plate

Number	Pin definition
1	Dry electrodes DRY+
2	Reference electrodes GND
3	Dry electrodes DRY-
4	PJ-342 dry electrode interface

Product information

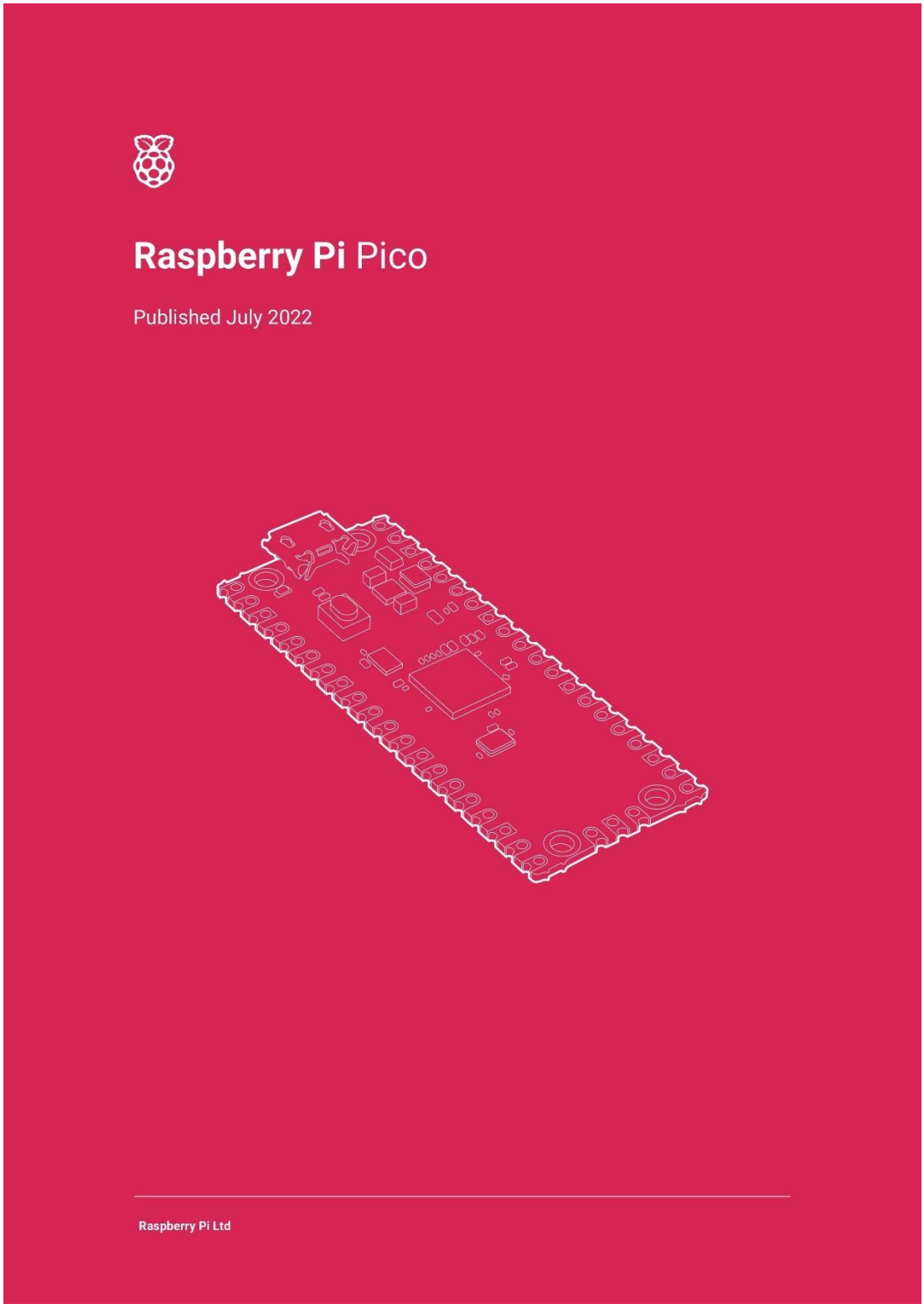
Product name	Dry electrode muscle sensor	Product name	Dry electrode guide plate
Picture		Picture	
Supply voltage	+3.3V ~ 5.5V	Electrode interface	PJ-342
Operating voltage	±3.0V	Length of line	50cm
Detection range	+/-1.5mV	Size	35×23mm
Electrode interface	PJ-342		
module interface	XH2.54-3P		
output range	0 ~ 3.0V		
Operating temperature	0 ~ 50℃		
Size	48×25mm		

Note: The power supply positive and negative poles must not be reversed, as this may cause damage to the module.

Packing list

Dry elecrode EMG Sensor Module *1, Electrode lead board *1 , Electrode line*1
Bandage*1,3P terminal line *1

C.2. TARJETA RASPBERRY PI PICO



Overview

Raspberry Pi Pico is the debut microcontroller-class board from Raspberry Pi. Built around our RP2040 silicon platform, Pico brings our signature values of high performance, low cost, and ease of use to the microcontroller space.

With a large on-chip memory, symmetric dual-core processor complex, deterministic bus fabric, and rich peripheral set augmented with our unique Programmable I/O (PIO) subsystem, RP2040 provides professional users with unrivalled power and flexibility. With detailed documentation, a polished MicroPython port, and a UF2 bootloader in ROM, it has the lowest possible barrier to entry for beginner and hobbyist users.

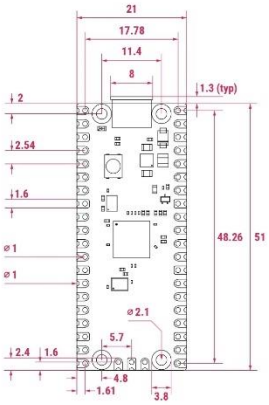
RP2040 is manufactured on a modern 40nm process node, delivering high performance, low dynamic power consumption, and low leakage, with a variety of low-power modes to support extended-duration operation on battery power.

Raspberry Pi Pico pairs RP2040 with 2MB of Flash memory, and a power supply chip supporting input voltages from 1.8-5.5V. It provides 26 GPIO pins, three of which can function as analogue inputs, on 0.1”-pitch through-hole pads with castellated edges. Raspberry Pi Pico is available as an individual unit, or in 480-unit reels for automated assembly.

Specification

Form factor:	21 mm × 51 mm
CPU:	Dual-core Arm Cortex-M0+ @ 133MHz
Memory:	264KB on-chip SRAM; 2MB on-board QSPI flash
Interfacing:	26 GPIO pins, including 3 analogue inputs
Peripherals:	• 2 × UART • 2 × SPI controllers • 2 × I2C controllers • 16 × PWM channels • 1 × USB 1.1 controller and PHY, with host and device support • 8 × PIO state machines
Input power:	1.8–5.5V DC
Operating temperature:	-20°C to +85°C
Production lifetime:	Raspberry Pi Pico will remain in production until at least January 2028
Compliance:	For a full list of local and regional product approvals, please visit pip.raspberrypi.com

Physical Specification



Note: all dimensions in mm

WARNINGS

- Any external power supply used with Raspberry Pi Pico shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment, and if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface, and should not be contacted by conductive items.
- The connection of incompatible devices to Raspberry Pi Pico may affect compliance, result in damage to the unit, and invalidate the warranty.
- All accessories used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

SAFETY INSTRUCTIONS

To avoid malfunction or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; Raspberry Pi Pico is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the corners to minimise the risk of electrostatic discharge damage.


Raspberry Pi Pico © 2020
39


Raspberry Pi
Raspberry Pi is a trademark of Raspberry Pi Ltd