FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

DISEÑO DE UN SISTEMA DE CONTROL DE UNA MANO ROBÓTICA MEDIANTE SEÑALES ELECTROMIOGRÁFICAS

AUTOR

JUAN DIEGO MANTILLA BRITO

AÑO

2019

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

DISEÑO DE UN SISTEMA DE CONTROL DE UNA MANO ROBÓTICA MEDIANTE SEÑALES ELECTROMIOGRÁFICAS

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de la Información.

Profesor Guía

MSc. David Fernando Pozo Espín

Autor

Juan Diego Mantilla Brito

Año

2019

DECLARACIÓN DEL PROFESOR GUÍA

“Declaro haber dirigido el trabajo, Diseño de un sistema de control de una mano robótica mediante señales electromiográficas, a través de reuniones periódicas con el estudiante Juan Diego Mantilla Brito, en el semestre 201920, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”

_____

David Fernando Pozo Espín

Máster en Automática y Robótica

C.I: 1717340143

DECLARACIÓN DEL PROFESOR CORRECTOR

"Declaro haber revisado este trabajo, Diseño de un sistema de control de una mano robótica mediante señales electromiográficas, del estudiante Juan Diego Mantilla Brito, en el semestre 201920, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Jorge Luis Rosero Beltrán

Máster en Ciencias con Especialidad en Automatización

C.I: 1803610185

DECLARACIÓN DE AUTORIA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”

_____

Juan Diego Mantilla Brito

C.I: 1715841233

AGRADECIMIENTOS

Agradezco en primer lugar a Dios porque desde que nací me ha bendecido con retos y oportunidades.

De manera especial, agradezco a mis padres que han sido mi fortaleza y motivación en esta travesía.

A mis profesores que a lo largo de este tiempo se convirtieron no sólo en instructores, sino también en amigos.

Y a las personas maravillosas y sinceras que llegaron a mi vida.

De todo corazón, MUCHAS GRACIAS.

DEDICATORIA

Quiero dedicar este trabajo y todos mis triunfos a mi madre porque a lo largo de mi vida nunca me han faltado sus atenciones, sus consejos, su amor y paciencia.

A mi padre, que siempre ha sido para mí un ejemplo de perseverancia y sacrificio por la familia.

Porque ellos siempre estarán a mi lado. Los amo.

RESUMEN

Las señales electromiográficas (EMG) son pulsos eléctricos que son emitidos por los músculos al realizar una acción. A lo largo del tiempo han sido objeto de estudio de considerables áreas de la ciencia, en especial en medicina y robótica que han podido crear soluciones tecnológicas para mejorar el estilo de vida de las personas.

En este trabajo de titulación se desarrolla un sistema embebido de adquisición y procesamiento de señales EMG que se pueden medir en el antebrazo derecho. El tratamiento de estas señales permite el control de un prototipo de mano robótica para abrirla o cerrarla.

Esto es posible mediante la utilización del dispositivo MYO Gesture Control Armband, el cual posee ocho sensores EMG superficiales que capturan y transmiten las señales musculares vía Bluetooth 4.0 al sistema de control compuesto principalmente por un microcontrolador ATmega328P y un STM32. Después, cada señal es procesada para extraer sus características (Valor Medio Absoluto, Varianza, Desviación Estándar y RMS) que son aplicadas en el algoritmo clasificador Naïve Bayes previamente entrenado por un grupo de siete usuarios. Luego se identifica la acción y se envía una señal de control a la mano robótica, para replicar el movimiento.

Finalmente en once personas, se realizan pruebas del sistema configurándolo para medir, encapsular y procesar las señales EMG en paquetes de diferentes tamaños para evaluar el comportamiento del sistema con los usuarios que formaron parte del entrenamiento y otros ajenos a este. Los resultados son tabulados y mediante matrices de confusión, se determina el porcentaje de exactitud del sistema que se encuentra entre 74,67% y 89,31% con tiempos de medición de 0,73 a 2,39 segundos y tiempos de procesamiento de 0,03 a 0,05 segundos. Mostrando un rango de tiempo total que está entre 0,76 y 2,44 segundos, lo que permite concluir que el tamaño del paquete EMG configurado para medir afecta directamente al tiempo de respuesta y a la exactitud de las predicciones.

ABSTRACT

Electromyographic (EMG) signals are electrical pulses that are emitted by the muscles when we perform an action. Over the time they have been object of study in considerable areas of science, especially in medicine and robotics that have been able to create technological solutions to improve the lifestyle of the people.

In this titling work, an integrated system is developed for the acquisition and processing of EMG signals that can be measured in the right forearm. The treatment of these signals allows the control of a robotic hand prototype to open or close it.

This is possible through the use of the MYO Gesture Control Armband device, which has eight surface EMG sensors that capture and transmit the muscle signals via Bluetooth 4.0 to the control system composed mainly of an ATmega328P microcontroller and an STM32. Then, each signal is processed to extract its features (Absolute Mean Value, Variance, Standard Deviation and RMS) that are applied in the Naïve Bayes classifier algorithm previously trained by a group of seven users. Then the action is identified, and a control signal is sent to the robotic hand, to replicate the movement.

Finally, with eleven people, system tests are performed configuring it to measure, encapsulate and process the EMG signals in packages of different sizes to evaluate the behavior of the system with the users that were part of the training and others outside it. The results are tabulated and by means of confusion matrices, we determine the system percentage of accuracy is between 74,67% and 89,31% with measurement times from 0,73 to 2,39 seconds and processing times of 0,03 to 0,05 seconds. Showing a total time range between 0,76 and 2,44 seconds, which allows us to conclude that the size of the EMG package configured to measure directly affects the response time and accuracy of the predictions.

ÍNDICE

1. INTRODUCCIÓN

"Cualquier tecnología suficientemente avanzada es equivalente a la magia" (Clarke, 2015). Con el pasar de los años, las ciencias encargadas de estudiar y desarrollar manos robóticas han evolucionado de forma notable. Desde diseños primitivos con operación limitada, hasta prototipos avanzados con capacidad de replicar acciones humanas con alta precisión. En la actualidad, existe una gran variedad de propuestas en el mercado que se diferencian generalmente en tecnología, comodidad y costo. Empresas e instituciones educativas a nivel mundial se encargan constantemente de experimentar métodos innovadores para el diseño y control de sistemas robotizados con el objetivo de construir partes humanas artificiales que trabajen del mismo modo que las naturales.

Como ejemplo de esto puede mencionarse que en el 2014, el Instituto de Tecnología de Harbin (China) construyó una mano adulta sintética compuesta por motores DC, una cabeza de engranajes, un tendón artificial y tres articulaciones para los dedos índice, medio, anular y meñique. Por otro lado, el pulgar fue diseñado con una articulación extra a los tres originales para permitir el movimiento por la palma, ofreciendo una gran precisión en el agarre de objetos (Liu, Feng, & Gao, 2014).

En cuanto a sistemas de control se tiene una comparación realizada en el 2013 por la Universidad Estatal de Idaho (Estados Unidos), entre las técnicas de control de Lógica Difusa (FL), Derivada Proporcional (PD) y el Sistema de Inferencia Neuro Difuso Adaptivo (ANFIS), que fueron aplicadas a una mano robótica con una libertad de catorce grados (Chen & Subbaram Naidu, 2013).

Dirigiéndose al mercado comercial de manos robóticas, este tiene varios modelos con diferentes características y costos. Por ejemplo, la marca DFRobot ofrece prototipos compuestos principalmente por cinco servomotores con capacidad de soportar hasta 500 gramos a un precio de 199 USD (DFRobot, 2019a). Así mismo, la empresa Custom Entertainment Solutions tiene a la venta una mano con una

libertad de 5 grados a un precio de 950 USD (Custom Entertainment Solutions, 2018). Finalmente, la compañía ROS Components posee un modelo con 20 grados de libertad a 90 000 Euros, un total de 119 700 USD (ROS Components, 2019).

A medida que las técnicas mecánicas y de control en manos robóticas han avanzado considerablemente, también lo ha realizado el estudio de la electromiografía, la cual ha brindado grandes aportes científicos en el campo de la robótica.

Haciendo síntesis, se puede definir a la electromiografía como la evaluación de las actividades eléctricas que son generadas por los músculos al momento de relajarse o contraerse. Gracias a estas señales, es posible desarrollar aplicaciones de robótica, medicina, domótica, entre otras (Shroffe & Manimegalai, 2013). A continuación de forma cronológica, se procede a mencionar algunos proyectos científicos que han utilizado estas señales, con la finalidad de poder entender su importancia.

En el 2014:

- Se logró innovar el área psiquiátrica al desarrollar un sistema portable construido a base de electrodos que adquieren los datos EMG al realizar una expresión en el rostro. Utilizando el algoritmo clasificador de red neuronal se pudo inferir el estado de ánimo del paciente, siendo un apoyo para el médico tratante (Gruebler & Suzuki, 2014).
- Se desarrolló una aplicación móvil orientado al ciclismo, misma que logra calcular en tiempo real la frecuencia cardíaca del deportista y su cadencia de pedaleo. Se complementa con sensores Shrimmer, los cuales capturan y transmiten señales ECG para la detección de latidos del corazón y EMG para determinar el nivel de presión en el pedal (Richer, Blank, Schuldhaus, & Eskofier, 2014).

En el 2015:

- Se realizó una comparación entre el sensor MYO Armband y un sensor EMG convencional. Para dicho estudio se diseñó un brazo robótico en Unity 3D que se comunicaba con estos periféricos mediante Bluetooth al computador. Al finalizar las pruebas, se concluyó que el MYO ofrecía un mejor funcionamiento mientras que se operaba con el mínimo esfuerzo posible (Shin, Ganiev, & Lee, 2015).

En el 2016:

- Se elaboró un sistema de adquisición de señales EMG basado en el cansancio muscular. Para ello se utilizó electrodos superficiales, los cuales amplifican, filtran y envían la señal a una aplicación que procesa la información por transformadas de Fourier con el fin de operar brazos robóticos (Correa-Figueroa, Morales-Sánchez, Huerta-Ruelas, González-Barbosa, & Cárdenas-Pérez, 2016).

En el 2018:

- Se desarrolló un proyecto innovador en el área de obstetricia el cual consistió en la captura y procesamiento de señales electrohisterográficas (EHG) las cuales son señales EMG producidas en las paredes uterinas. Estas fueron analizadas por varios clasificadores entrenados que permitían identificar entre pulsos normales del embarazo y contracciones de parto. Obteniendo gran precisión en sus resultados (Tsipouras, 2018).
- Se produjo un estudio sobre la enfermedad de Parkinson, el cual consistió en la lectura de señales EMG de los músculos de las piernas, con la finalidad de observar cómo evoluciona esta afección y poder establecer biomarcadores cuantitativos (Flood, Jensen, Malling, & Lowery, 2018).

En el 2019:

- Se desarrolló un juego controlado por señales EMG centrado en la rehabilitación de personas que hayan sufrido accidentes cerebrovasculares. Dicho juego se desarrolló en Matlab y consistía en superar ciertos niveles que requerían movimientos precisos de la mano con la finalidad de rehabilitar movilidad (Ghassemi et al., 2019).
- Se diseñó una aplicación móvil multicapa para dispositivos Android capaz de procesar la información EMG capturada e indicar el estado muscular de la persona. El proyecto tenía como finalidad hacer portable la lectura y procesamiento EMG (Karimpour, Parsaei, Sharifian, Rojhani, & Yazdani, 2019).

Tras los ejemplos anteriores, se puede concluir que, en los últimos años se ha conseguido aprovechar en gran medida las señales EMG para alcanzar diversos propósitos.

Actualmente, existen algunos equipos especializados en la lectura EMG. El más popular es el sensor MYO Armband, el cual de forma general, se define como un periférico utilizado para comunicarse con el computador y controlar ciertas funcionalidades de la computadora como el puntero del ratón, presentaciones, juegos, entre otras. Posee una gran perspectiva a futuro gracias a que es orientado a todo público por su facilidad de uso (Rawat, Vats, & Kumar, 2016).

Thalmic Labs, su empresa creadora, se encarga constantemente del desarrollo de aplicaciones que exploten el potencial del sensor. La más popular es “MYO Diagnostics”, en la cual se puede observar la actividad muscular del antebrazo en una interfaz gráfica, siendo de gran ayuda en la fisioterapia ya que se puede comprender de mejor manera el sistema de miocardio del paciente (Sathiyanarayanan & Rajan, 2016).

Resumiendo, su funcionamiento desde una perspectiva técnica experimental, el MYO Armband captura 8 señales electromiográficas, las cuales pueden ser

interpretadas y transformadas en estadísticas como: Valor Absoluto Medio (MAV), Varianza (VAR), Amplitud de Willinson (WAMP), Longitud de Forma de Onda (WL), entre otros. Ciertos autores, tras realizar varias investigaciones, han concluido que el MAV es la mayor recomendación para la extracción de características y aplicación en algoritmos clasificadores (Arief, Sulistijono, & Ardiansyah, 2015).

En cuanto a algoritmos clasificadores, se puede destacar que en el 2017 se realizó un estudio donde se comparó varios clasificadores de señales EMG, entre los cuales estuvieron el algoritmo de redes neuronales de avance (FFN), máquinas de vectores de soporte (SVM), el clasificador Naïve Bayes (NBC) y el análisis discriminante lineal (LDA). En dicho estudio de experimentó con el algoritmo NBC utilizando el MYO Armband en el área del antebrazo de ocho participantes para identificar diversos gestos de la mano. Los resultados de las pruebas fueron óptimos tanto en reconocimiento como en tiempo de respuesta (Morales & Pozo, 2017).

Así mismo, en los últimos años se han presentado algunos proyectos de implementación de clasificadores orientados a la interpretación de reconocimiento gestual. A continuación, se resume los tres más relevantes:

En el 2014, se utilizó el clasificador SVM (máquinas de soporte vectorial) en un sistema con un sensor Leap Motion, en donde se extraían los parámetros de orientación y posición de las yemas de los dedos que posteriormente, eran analizadas por este algoritmo. El proyecto también se complementaba con un Kinect que obtenía la profundidad de la mano, obteniendo una gran precisión en la identificación del movimiento (Marin, Dominio, & Zanuttigh, 2014).

En el 2015, se aplicó el clasificador de redes neuronales en un sistema de reconocimiento gestual 3D aplicado al área automotriz con el objetivo de que el usuario pueda controlar diversos aspectos del vehículo como el aire y audio utilizando gestos (Molchanov, Gupta, Kim, & Kautz, 2015).

Igualmente en el 2015, se realizó un estudio de los clasificadores más utilizados como SVM, redes neuronales, arboles aleatorios y vecinos cercanos aplicados en una mano 3D virtual. Este trabajo se hizo con la finalidad de comparar el desempeño de cada algoritmo en el reconocimiento gestual (Cheng, Yang, & Liu, 2015).

En el presente proyecto de titulación se procederá a utilizar el MYO Armband en el área del antebrazo para controlar una mano robótica mediante recolección de señales EMG y su procesamiento aplicando un algoritmo clasificador.

1.1. Objetivo General

Diseñar un sistema de adquisición y procesamiento de señales electromiográficas (EMG) para realizar el control de una mano robótica, utilizando un algoritmo clasificador.

1.2. Objetivos Específicos

- Realizar un estudio de las señales EMG y el funcionamiento del sensor MYO Armband.
- Adquirir las señales EMG del MYO Armband y realizar el procesamiento para la obtención de sus características.
- Aplicar un algoritmo clasificador para identificar movimientos de la mano humana, mediante señales EMG.
- Realizar las pruebas de funcionamiento para el control del prototipo de mano robótica.

1.3. Alcance

El alcance de este trabajo de titulación es realizar la adquisición, procesamiento y análisis de 8 señales electromiográficas del sensor MYO Armband que estará

ubicado en el área del antebrazo derecho, para manejar una mano robótica, mediante un sistema microcontrolado.

Para llegar a cumplir con lo mencionado anteriormente, se procesará las señales entrantes en el sistema microcontrolado y se obtendrá al menos cuatro características de cada una, como valor medio, varianza, desviación estándar y RMS.

Posteriormente, se hará uso del clasificador Naïve Bayes para reconocer al menos dos gestos de la mano: abierto y cerrado en base al entrenamiento del sistema.

Finalmente, se procederá a hacer pruebas de reconocimiento y control de la mano robótica comercial “DFRobot - Bionic Robot Hand (Right)” (DFRobot, 2019a) y se analizará la fiabilidad del sistema.

1.4. Justificación

Basándose en los antecedentes presentados anteriormente, el área científica encargada de experimentar con manos robóticas o virtuales ha tenido un crecimiento elevado en los últimos años.

Sin embargo, casi todos los estudios realizados, ya sea con señales electromiográficas o con cualquier otro tipo de control, necesitan de un computador como intermediario entre el prototipo y el usuario, o en caso de que no lo requieran, estos adquieren tiempos de respuesta muy lentos, produciendo que el rendimiento no sea el adecuado.

Este proyecto pretende establecer una comunicación directa entre usuario y mano robótica mediante un sistema microcontrolado y con el menor tiempo posible de respuesta y procesamiento.

2. MARCO TEÓRICO

En el presente capítulo se explica de forma detallada y ordenada todos los conceptos teóricos relacionados con las manos robóticas, señales electromiográficas (EMG), el sensor MYO Armband, algoritmos clasificadores y otros temas claves que aportan para una mejor comprensión del desarrollo de este trabajo de titulación.

2.1. Manos robóticas

Una mano robótica es definida como un sistema mecatrónico conformado por sensores, motores, y circuitos electrónicos. Están integradas en una estructura mecánica construida con diversos materiales que replican la estructura de los tendones humanos. Realizan acciones naturales como agarrar e interactuar con objetos, abrir puertas, entre otras. Con la finalidad de aplicarse en proyectos de prótesis inteligentes o robots humanoides con IA (Ryan, Metz, & Taylor, 2018).

Generalmente, las manos robóticas pueden clasificarse en dos tipos. El primero, como se puede observar en la Figura 1, corresponde a aquellas que son diseñadas comúnmente con propósitos industriales tales como el ensamblaje de autos. Estas manos tienen una forma de “pinza” de dos o tres dedos y vienen integradas en brazos robóticos de gran fuerza (Levin, 2018).

El segundo tipo se relaciona con el estudio científico entre la mecánica, electrónica e incluso neurociencia para diseñar manos robóticas que simulen el comportamiento de las humanas. Estas manos, como se puede observar en la Figura 2, poseen articulaciones artificiales que son denominadas “grados de libertad” o DoF por sus siglas en inglés (Degrees of Freedom) unidos a actuadores intrínsecos y extrínsecos correspondientes a los músculos. (Controzzi, Cipriani, & Carrozza, 2014).

Figura 1. Mano robótica tipo “pinza”.

Tomado de (ROBOTIQ, 2019)

Figura 2. Mano robótica humanoide.

Tomado de (SCHUNK, 2019).

2.1.1. Modelos comerciales

Actualmente, existe una gran cantidad de empresas desarrolladoras de sistemas robotizados las cuales ofrecen prototipos que se diferencian ya sea por su número de articulaciones o grados de libertad, así como su peso, tecnología u otras características.

A continuación, se detalla seis modelos ofertados en el mercado.

2.1.1.1. Shadow Dexterous Hand

Es un prototipo avanzado con 20 DoF que pretende replicar en gran medida la libertad de una mano humana lo más preciso posible, como se observa en la Figura 3.

Posee alrededor de 129 sensores de posición, fuerza, presión, entre otros. Adicionalmente, está abierto a cualquier software de control y puede sostener hasta 4 Kg. Con un peso de 4.3 Kg, este modelo tiene un costo en el mercado alrededor de 119 700 USD (ROS Components, 2019).

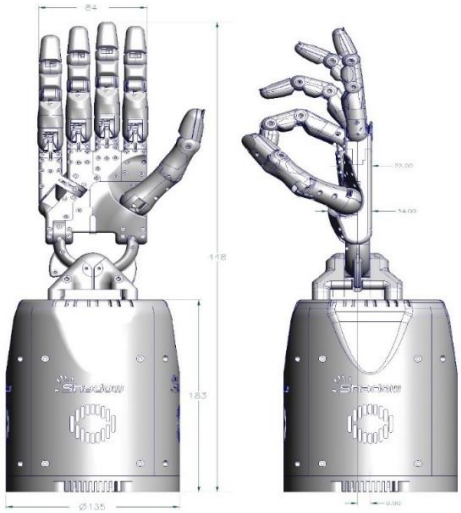


Figura 3. Shadow Dexterous Hand.

Tomado de (ROS Components, 2019).

2.1.1.2. DYN Hand GEN3

Es un diseño recubierto con piel sintética el cual se acerca a una mano real, como se puede apreciar en la Figura 4. Tiene 13 grados de libertad y su esqueleto es de acero inoxidable y bronce. Puede ser controlado por USB o TTL. Con un peso de 6 Kg, este prototipo está a la venta por un precio de 15 550 USD (Custom Entertainment Solutions, 2019).

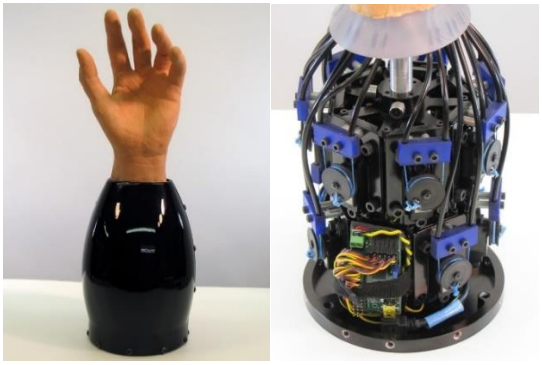


Figura 4. DYN Hand GEN3

Tomado de (Custom Entertainment Solutions, 2019)

2.1.1.3. AR10 Humanoid Robot Hand

Es un modelo el cual posee 10 DoF y su mecanismo de movimiento se basa en el uso de motores lineales, como se indica en la Figura 5. Está construida con materiales de aluminio y plástico.

Además, es un prototipo que permite personalización, es decir, es posible instalar dedos propios impresos en 3D e incluso tiene cuatro entradas analógicas disponibles para agregar sensores. La desarrolladora, Active 8 Robots, tiene de venta este modelo en 4 252,54 USD (Active 8 Robots, 2019).

Figura 5. AR10 Humanoid Robot Hand.

Tomado de (Active 8 Robots, 2019).

2.1.1.4. Youbionic Hand 2019

Es un prototipo con el armazón impreso completamente en 3D, que tiene un sistema de movimiento apoyado por motores lineales “Actuonix PQ12”, ligeros (15 gramos cada uno) e independientes que brindan 6 DoF, como se muestra en la Figura 6.

Se puede complementar fácilmente con una placa Arduino y puede sostener hasta 4,6 Kg. Este modelo está a la venta con la empresa desarrolladora a 1 295,00 USD (Youbionic, 2019).

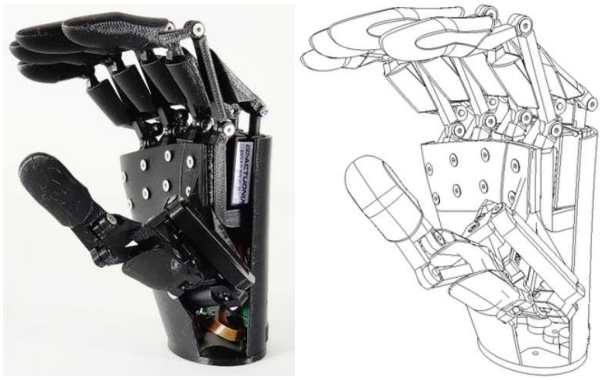


Figura 6. Youbionic Hand 2019

Tomado de (Youbionic, 2019).

2.1.1.5. Mecha X Robotic Hand

Esta mano robótica que ofrece 5 DoF, está diseñada principalmente con acero inoxidable y partes impresas en 3D, tal y como se observa en la Figura 7. Su estructura está desarrollada para soportar errores de control, por ejemplo, que los dedos se doblen hacia atrás.

Con un peso de 2 Kg, se encuentra a la venta por 950 USD (Custom Entertainment Solutions, 2018).

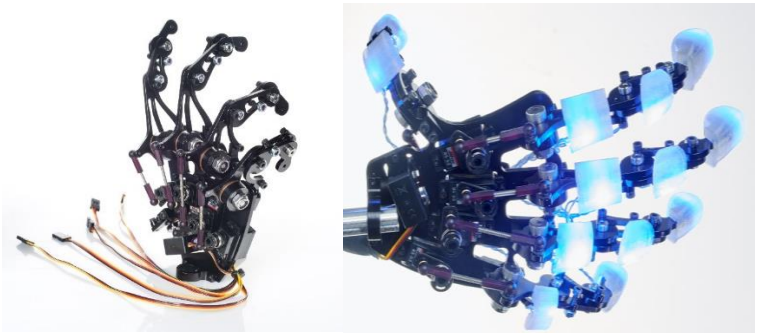


Figura 7. Mecha X Robotic Hand.

Tomado de (Custom Entertainment Solutions, 2018).

2.1.1.6. Bionic Robot Hand (Right)

Este modelo es el más económico en el mercado actual ya que su estructura es diseñada en acrílico la cual le da un peso de 916 gramos como se puede observar en la Figura 8 (DFRobot, 2019a).

El mecanismo de movimiento es realizado con servomotores convencionales combinados con sistemas independientes de resortes los cuales expanden o contraen los dedos, así como se indica en la Figura 9.

Con capacidad de sostener hasta 500 gramos y con 5 grados de libertad (DoF), esta mano robótica se encuentra en el mercado a 199 USD (DFRobot, 2019a).

Debido a su precio conveniente, sus costos reducidos en repuestos y a su fácil adaptabilidad a cualquier sistema microcontrolado, esta mano robótica será la utilizada para el presente proyecto de titulación, en la cual se implementará el sistema de control por señales electromiográficas.

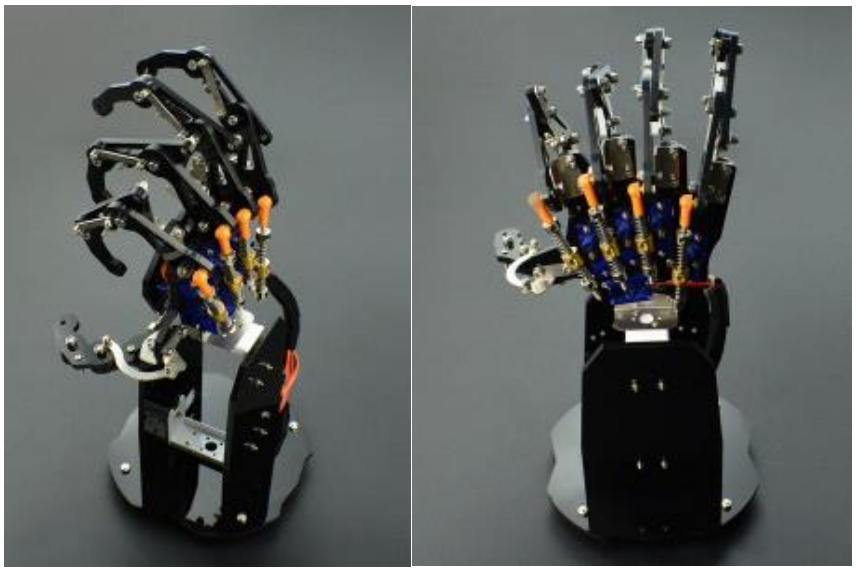


Figura 8. Bionic Robot Hand (Right).

Tomado de (DFRobot, 2019a).

Figura 9. Mecanismo de resortes de la “Bionic Robot Hand (Right)”.

Tomado de (DFRobot, 2019a).

2.2. Señales electromiográficas (EMG)

En la biomedicina, las señales EMG son definidas como corrientes eléctricas producidas por el ser humano al momento de relajar o contraer un músculo, que pueden ser interpretadas en valores numéricos, gráficos o sonidos. Anatómicamente, estas señales son producidas por el sistema nervioso ya que este es el encargado de controlar la actividad fisiológica y anatómica del cuerpo (Gokgoz & Subasi, 2015).

Por esta razón, en un inicio fueron utilizadas en la neurociencia para poder diagnosticar enfermedades o trastornos neuromusculares por medio de exámenes electromiográficos (Mishra, Bajaj, Kumar, Sharma, & Singh, 2017).

Estos generalmente son realizados por medio de electrodos de Ag/AgCl, que están compuestos por un núcleo de plata (Ag) rodeado por un revestimiento de cloruro de plata (AgCl) que los hace inmunes a la sal y al agua corporal. Acompañados de amplificadores y filtros, estos electrodos permiten obtener un muestreo y descomposición óptima en tramas individuales llamadas “potencial de acción de la unidad motora” (MUAPTs), como se indica en la Figura 10 (Karlik, 2014).

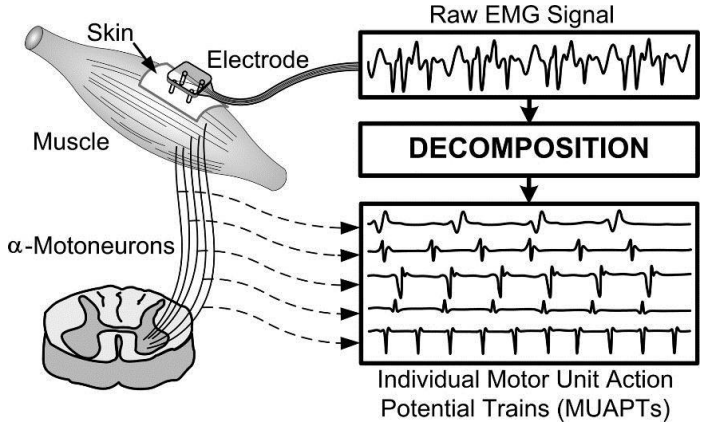


Figura 10. Esquema del proceso de captura y descomposición de señales EMG.

Tomado de (Barrios, 2014).

Sin embargo, se debe destacar que las señales electromiográficas, pueden llegar a deteriorarse por el ruido ambiental e incluso cuando estas pasan a través de tejidos. Razón por la cual, cada persona tiene su identidad electromiográfica particular que se define de acuerdo con su contextura física (Gokgoz & Subasi, 2015).

2.2.1. Sensores de lectura electromiográfica

Según varios autores, se tiene una cifra genérica de 600 músculos en el cuerpo humano ya que, esto depende de la agrupación que se realice al contar varios músculos como uno solo o no. Por lo cual, este número varía de autor en autor (McGuinness, 2018). Cada músculo emite sus propias señales EMG y gracias a esto, los campos de investigación y medicina han desarrollado estudios importantes que han sido posible gracias a los diversos equipos de lectura e interpretación electromiográfica existentes. El uso de electrodos es un modelo de lectura superficial muy común que se complementa con ciertos sensores que filtran y entregan una señal EMG limpia, algunos de ellos se expondrán a continuación.

2.2.1.1. MyoWare Muscle Sensor

Es un sensor con un tamaño equivalente a una moneda de 25 centavos, compuesto por tres electrodos, dos de ellos miden las señales desde la mitad hasta el final del músculo y el último es de referencia, como se muestra en la Figura 11. Este sensor captura una señal electromiográfica, filtra gran parte del ruido, las rectifica y finalmente las amplifica hasta generar una sola señal analógica que pueda interpretarse por un sistema microcontrolado (Advancer Technologies, 2019).

Como otras características, se debe destacar que este sensor no requiere de un voltaje de entrada externo, sino que puede ser conectado a la alimentación del microcontrolador que se utilice y su señal analógica ofrece una mejor captura de la señal. Se lo utiliza mayormente en los bíceps y solo puede tomar correctamente la

señal cuando se hace gran fuerza en el movimiento. Se lo puede encontrar en el mercado por un precio alrededor de 37,95 USD (Pololu, 2019).

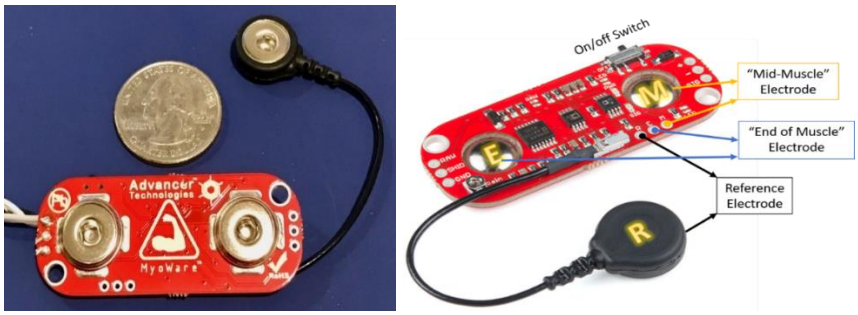


Figura 11. MyoWare Muscle Sensor – Tamaño y Diseño.

Tomado de (Medium, 2018).

2.2.1.2. Bitalino Electromyography (EMG) Sensor

Este sensor electromiográfico se caracteriza principalmente por permitir trabajar con tres electrodos ya sean secos o con gel conductor. Razón por la cual, no vienen incluidos en el sensor, como se muestra en la Figura 12. Además, tiene un modo de operación bipolar en donde no importa en donde se encuentren los electrodos, estos van a realizar la medición de señales EMG, filtrado y rectificación. Se puede adaptar a cualquier sistema Arduino ya entrega una señal analógica limpia y lista para ser procesada. Debe tener un voltaje de entrada externo de 2 V a 3.5 V, y está en el mercado a 25,46 USD (Bitalino, 2019).

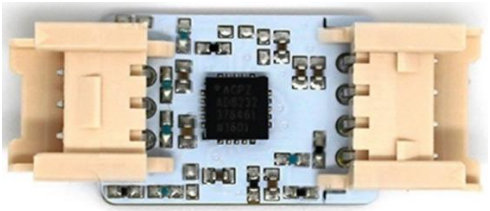


Figura 12. Bitalino Electromyography (EMG) Sensor.

Tomado de (Bitalino, 2019).

2.2.1.3. Grove - EMG Detector

Grove es un sensor electromiográfico con tres electrodos que toman una ligera señal EMG, la amplifica y la filtra para dar como resultado una señal analógica que puede ser trabajada por cualquier sistema Arduino.

Se puede considerar a este sensor como uno básico para trabajos iniciales con señales EMG, por lo cual no es recomendado para aplicaciones médicas.

No requiere de alimentación externa y los electrodos no se encuentran embebidos, como se puede observar en la Figura 13. Finalmente, tiene un costo de 29,90 USD (Seeed Studio, 2019).

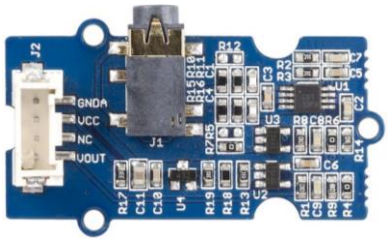


Figura 13. Grove - EMG Detector.

Tomado de (Seeed Studio, 2019).

2.2.1.4. Gravity: Analog EMG Sensor by OYMotion

Gravity es un dispositivo de lectura electromiográfica diseñada por DFRobot y OYMotion, cuyo objetivo fue crear un sensor económico que realice la captura EMG de forma completa, como se instancia en la Figura 14. Mediante circuitos de electrodos, filtros, amplificadores y rectificadores, se toma una señal EMG débil y se la amplifica 1000 veces mientras que se filtran los ruidos.

Como salida se tiene una señal analógica en la cual su intensidad es relacionada con la fuerza del movimiento. Generalmente, este sensor fue diseñado para medir

fuerza muscular, tensión en el músculo y análisis de actividades neuronales, que se puedan aplicar para actividades de control, mas no médicas. Trabaja normalmente con cualquier placa Arduino, y está a un precio de 37,50 USD (DFRobot, 2019b).

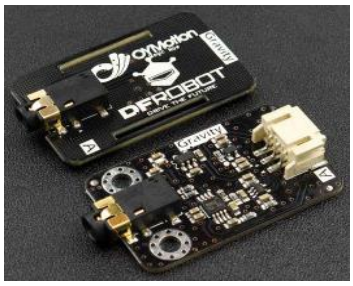


Figura 14. Gravity: Analog EMG Sensor by OYMotion.

Tomado de (DFRobot, 2019b).

Una vez expuesto los ejemplos anteriores, es importante mencionar que estos modelos son únicamente sensores que capturan de dos a tres señales electromiográficas, por lo cual pueden considerarse “básicos” ya que como se indicó anteriormente, el cuerpo humano tiene una gran variedad de músculos los cuales emiten señales EMG propias.

El procesamiento de dos o tres señales en una área que tiene varios músculos, como por ejemplo el antebrazo que posee alrededor de 18 (Health Line, 2018), puede generar resultados poco prometedores cuando se analiza un movimiento.

Por lo cual, para el desarrollo de este trabajo de titulación se buscó un sensor que puede capturar más de tres señales para interpretar una acción humana de forma correcta.

2.3. MYO Gesture Control Armband

El dispositivo MYO Armband es definido básicamente como un periférico comercial que adquiere, filtra y amplifica ocho señales electromiográficas provenientes del

área superior del antebrazo al realizar algún movimiento y posteriormente las transmite hacia un módulo bluetooth 4.0 (incluido en el paquete de venta como se puede observar en la Figura 15).

El dispositivo se conecta al computador con el objetivo de controlar ciertas aplicaciones como PowerPoint, Prezi, YouTube, o inclusive para controlar el puntero y un teclado en pantalla.

Adicionalmente, la empresa desarrolladora Thalmic Labs, tiene una serie de aplicaciones propias para poder observar la actividad EMG del usuario (MYO Diagnostics) e incluso, este dispositivo es utilizado en varios juegos de computadora o control de drones (Rawat et al., 2016).

Figura 15. MYO Gesture Control Armband.

Tomado de (Stern, 2018).

2.3.1. Estructura externa

Partiendo desde el exterior, el sensor MYO Armband está dividido en ocho secciones plásticas, las cuales contienen los elementos electrónicos de captura, procesamiento y alimentación distribuidos entre sí. Estas partes están unidas por medio de un plástico expandible el cual permite que el sensor se adapte fácilmente a cualquier tipo de antebrazo, como se puede observar en la Figura 16

(Sathiyanarayanan, Mulling, & Nazir, 2015). Continuando con la estructura, MYO Armband posee un puerto de carga vía USB que le permite recargar sus dos baterías internas de 3,7 V a 260mAh, como se muestra en la Figura 17 (Stern, 2018).

Adicionalmente, se tiene el logotipo de Thalmic Labs de color azul el cual según su iluminación da a conocer si el MYO Armband está conectado y sincronizado (mediante un gesto predeterminado) a su módulo Bluetooth; o si se encuentra en reposo. Por último, se tiene un LED de estado que presenta varios colores para indicar la situación actual del MYO Armband, mismos que se describen de mejor manera en la Figura 18 y en la Tabla 1.

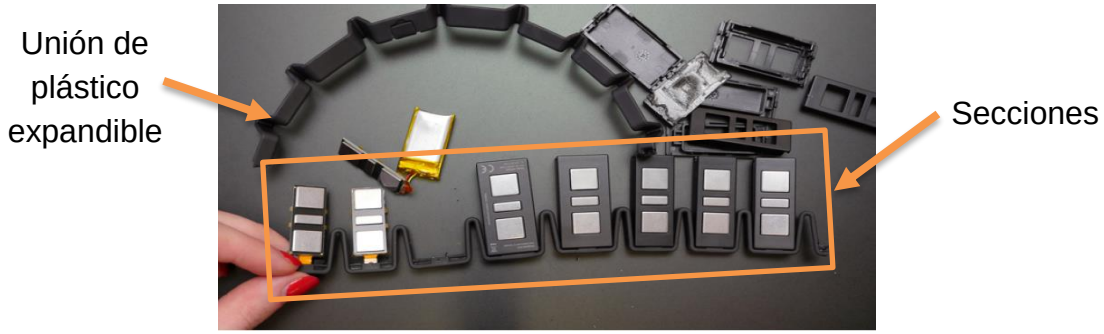


Figura 16. Secciones del MYO Armband.

Modificado de (Stern, 2018).

Figura 17. Batería del MYO Armband.

Tomado de (Stern, 2018).

Figura 18. Indicadores y puerto de carga del MYO Armband.

Modificado de (Mahmoud Abduo & Galster, 2015).

Tabla 1.

Estados del MYO Armband.

Color LED	Actividad Logotipo	Descripción
Naranja intermitente	Logotipo apagado.	El sensor se está cargando.
Verde continuo	Logotipo apagado.	El sensor se encuentra cargado.
Naranja intermitente	Logotipo encendido.	Se está acabando la batería.
Ningún color.	Logotipo apagado.	El sensor se encuentra apagado.
Azul permanente.	Logotipo encendido.	El sensor se encuentra conectado y sincronizado al módulo bluetooth.
Azul permanente.	Logotipo intermitente.	El sensor se encuentra conectado, pero no se ha realizado el gesto de sincronización.
Ningún color.	Logotipo encendido.	El MYO no está conectado.

Ningún color.	Logotipo intermitente.	El sensor no está conectado ni sincronizado.
Azul permanente.	Logotipo intermitente (rápido).	El sensor se está ambientando a la piel del usuario.
Morado intermitente.	Logotipo apagado.	Se actualizó el firmware.
Rojo intermitente.	Logotipo apagado.	Ha ocurrido algún error de firmware.
Rojo permanente.	Logotipo apagado.	Ha ocurrido algún error de hardware.

Adaptado de (Support GetMyo, 2019).

2.3.2. Estructura interna

Internamente, el dispositivo MYO Armband está compuesto por ocho electrodos superficiales secos de acero inoxidable, los cuales están conectados entre sí por un material PCB flexible insertado en la banda que une las secciones del MYO. Es importante mencionar que el MYO Armband técnicamente es un conjunto de sensores que son controlados por una placa particular localizada en la sección que posee el logotipo y el LED indicador.

Dicha placa principal está constituida además por un motor de vibración, un módulo Bluetooth, un conector USB y un procesador de 32 bits de bajo consumo energético “Kinetis MK22FN1M0” tipo ARM Cortex M4, como se puede observar en la Figura 19. Este procesador posee las siguientes características destacadas:

- Conversión analógica a digital flexible de 16 bits.
- 10 modos de operación de potencia.
- 1 Mb de memoria Flash.
- 128 Kb de memoria ROM.

Figura 19. Placa principal del MYO Armband.

Modificado de (Stern, 2018).

Por otra parte, las dos baterías de 3,7 V a 260mAh mencionadas anteriormente, se encuentran en las dos secciones adyacentes a la principal. Finalmente, cada electrodo es conectado a un módulo STM78544IC, el cual amplifica y filtra parte de la señal para ser entregada a la sección principal, como se puede apreciar en la Figura 20 (Mannion, 2016).

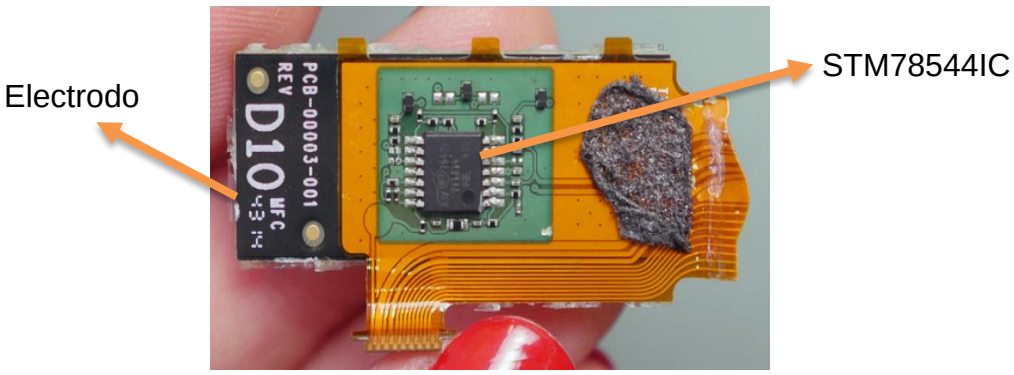


Figura 20. Placa principal del MYO Armband.

Modificado de (Stern, 2018).

2.3.2.1. Unidad de Medición Inercial

- La IMU (Unidad de Medición Inercial) del MYO Armband, está compuesta por un giroscopio de tres ejes, un acelerómetro de tres ejes y un magnetómetro

de tres ejes. Los cuales tiene como objetivo adquirir la orientación y posición relativa del antebrazo, a una frecuencia de muestreo de datos de 50 Hz.

- La orientación es calculada en términos de balanceo e inclinación del MYO Armband.
- La velocidad angular es obtenida en formato vectorial, mientras que la aceleración es medida por el acelerómetro (Mahmoud Abduo & Galster, 2015).

2.3.3. Lectura electromiográfica

La lectura EMG realizada por el MYO Armband, es un proceso que involucra sus ocho sensores (también denominados “canales”) que trabajan de forma simultánea para obtener señales eléctricas que viajan en músculos específicos del antebrazo a una frecuencia de muestreo de 200 Hz. Por otro lado, la IMU adquiere los datos de posición y orientación los cuales complementan la información del movimiento y son transmitidas por el módulo Bluetooth incorporado. Es importante resaltar que la señal electromiográfica inicial es analógica, pero para su transmisión hacia el terminal se convierte en datos digitales, como se puede observar en el ejemplo de la Figura 21 (Montoya-Leal et al., 2017).

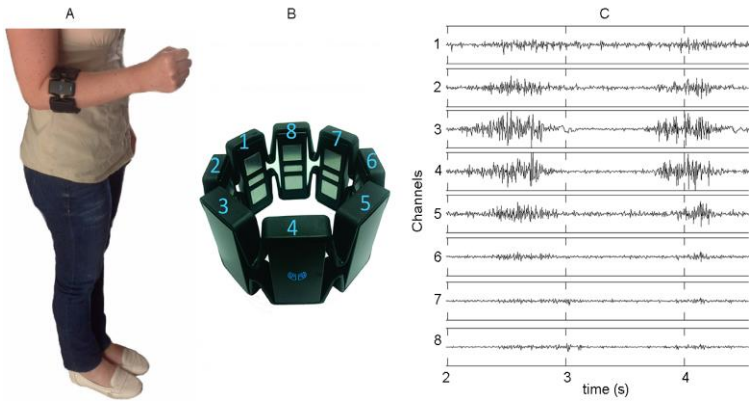


Figura 21. Ejemplo de señales electromiográficas obtenidas por el MYO Armband.

Tomado de (Montoya-Leal et al., 2017).

2.3.3.1. Músculos involucrados

Como se mencionó anteriormente, el MYO Armband recoge señales EMG de distintos músculos, mismos que se pueden apreciar en la Figura 22.

A continuación, se explica brevemente sobre estos:

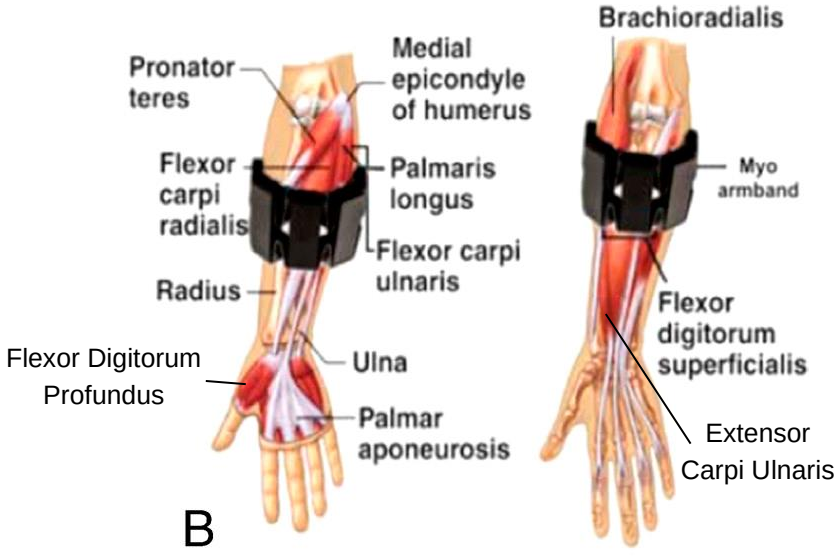


Figura 22. Músculos involucrados en la lectura EMG del MYO Armband.

Modificado de (Visconti et al., 2018).

1. **Palmaris Longus:** Está ubicado desde el codo y avanza hasta la mitad del antebrazo, en donde tiene como función apoyar a las articulaciones de la muñeca (Health Line, 2015g) (Visconti, Gaetani, Zappatore, & Primiceri, 2018).
2. **Flexor Carpi Radialis:** Es un músculo delgado superficial que se lo puede encontrar al inicio del antebrazo hasta la base del índice, y es aquel que se puede observar cuando la muñeca se flexiona. De igual manera, apoya en la flexibilidad de la articulación (Health Line, 2015c) (Visconti et al., 2018).

3. **Brachioradialis:** Ubicado en el antebrazo, es el encargado de flexionar el codo y de mantener la mano en hacia arriba o hacia abajo cuando esta realiza alguna de esas acciones (Health Line, 2015a) (Visconti et al., 2018).
4. **Pronator Teres:** Se lo puede encontrar debajo del codo y tiene como función principal, permitir el movimiento de giro del antebrazo (Health Line, 2015h) (Visconti et al., 2018).
5. **Flexor Digitorum Superficialis:** Ubicado en el centro del antebrazo, este músculo es un responsable extrínseco del movimiento de los dedos índice, medio anular y meñique (Health Line, 2015f) (Visconti et al., 2018).
6. **Flexor Carpi Ulnaris:** Se lo encuentra desde el húmero y avanza hasta la base del meñique, pasando por los huesos de la muñeca. Permite la flexión de la muñeca en función del húmero (Health Line, 2015d) (Visconti et al., 2018).
7. **Flexor Digitorum Profundus:** Comienza en el antebrazo y termina en las falanges de las puntas de los dedos y tiene como función ayudar en la flexión de los cinco dedos. Ciertos autores lo consideran como un músculo de la mano, mas no del antebrazo (Health Line, 2015e) (Morales & Pozo, 2017).
8. **Extensor Carpi Ulnaris:** Está ubicado desde el húmero y se extiende hasta la base del meñique. Tiene como función apoyar en el movimiento lateral de la muñeca (Health Line, 2015b) (Morales & Pozo, 2017).

2.3.3.2. Eventos y gestos predeterminados

Posterior a la lectura EMG y captura IMU, en la sección principal del MYO Armband se realiza un procesamiento de estos elementos que dan como resultado información del movimiento realizado (también denominada “información gestual”) e información espacial, que son enviadas hacia la aplicación de computadora a manera de eventos que se pueden dividir en tres clases, expuestas a continuación.

- **Eventos gestuales:** Es la información de la pose o gesto de la mano en ese instante. Suceden de forma irregular, es decir, el MYO Armband sólo

detectará un cambio de evento cuando la persona modifique el gesto a algún otro que este ya tenga en su configuración.

- **Eventos espaciales:** Es la información de orientación del antebrazo y la mano. Suceden de forma regular, es decir, el MYO Armband detecta cambios continuos que dependen de la velocidad en la que se esté realizando alguna acción.
- **Eventos auxiliares:** Es información propia del MYO Armband para hacer conocer al usuario que ha ocurrido alguna novedad. Suceden cuando se indica al usuario se conectó o se desconectó el módulo Bluetooth, o cuando este está a punto de entrar en reposo.

Es importante mencionar que gracias a estos eventos, el usuario puede colocarse el sensor en el brazo derecho o izquierdo sin que esto afecte al funcionamiento.

Esto es debido a que el MYO Armband tiene un movimiento de sincronización predefinido el cual al ser realizado, identifica si se lo está usando en la mano derecha o izquierda y lo confirma mediante una vibración (Sathiyanarayanan et al., 2015).

Los gestos reconocidos por defecto por este sensor se los puede observar en la Figura 23 y son:

1. Abierto (Open).
2. Palma vertical hacia atrás, es el movimiento de sincronización (Wave out).
3. Palma vertical hacia adelante (Wave in).
4. Cerrado o puño (Fist).
5. Pellizco (Pinch).

Figura 23. Gestos reconocidos por defecto en el MYO Armand.

Tomado de (Tatarian, Parente, Couceiro, & Faria, 2018).

Como ya se mencionó, el dispositivo MYO Armband fue diseñado inicialmente como un periférico del computador. Sin embargo, la atención hacia este sensor ha crecido tanto que en la actualidad se lo investiga para otros propósitos como la robótica, control de drones, etc. Esto se debe a que si bien es cierto que tiene un sistema embebido de procesamiento e identificación de gestos, sigue siendo un dispositivo electrónico que puede entregar datos EMG limpios a otro sistema microcontrolado que posea un módulo Bluetooth 4.0 con una configuración especial.

Posteriormente, dicho sistema microcontrolado debe ser capaz procesar los datos electromiográficos entregados y obtener ciertos parámetros o características que finalmente deben ser analizadas por un algoritmo clasificador que determinará en un porcentaje qué gesto se está realizando, para propósitos de control. En este trabajo de titulación, se procederá a utilizar el algoritmo clasificador Naïve Bayes el cual se explicará a continuación.

2.4. Algoritmos clasificadores

En el área de machine learning, un algoritmo clasificador puede ser descrito como una serie de ecuaciones aplicadas a la información (features) proporcionada por un

objeto que realiza una acción (evento), con el objetivo de poder relacionar la misma mediante un porcentaje, a una acción puntual (clase) previamente configurada (entrenada) a manera de predicción (Aggarwal, 2014).

Partiendo de esta definición, a continuación se explica de manera detallada cada elemento relacionado al concepto previo.

2.4.1. Características o features

Una característica o feature es definida como una particularidad variable que tiene un objeto el cual será aplicado en un algoritmo clasificador para poder identificar si este pertenece a una clase determinada. Puede ser representada de forma numérica o gráfica (Aggarwal, 2014).

Vinculando esta definición con el proyecto de titulación, las características son propiedades estadísticas obtenidas por cada segmento de señal electromiográfica entregada por el MYO Armband; mientras el usuario realiza una acción con la mano, también denominado “evento”.

2.4.2. Etiquetas de clases

En machine learning, una clase es definida como un grupo de características fijas las cuales sirven como modelo de comparación entre las que son variables. En relación con el tema de titulación, una clase corresponde a la información del movimiento abierto y cerrado de la mano humana.

Es importante mencionar que para la construcción de estas clases es necesario realizar un “entrenamiento”, el cual consiste en la captura de las características electromiográficas de una o varias personas realizando el movimiento deseado y posteriormente, procesar la información matemáticamente como si fuese una sola la cual que será configurada de forma manual en el algoritmo clasificador (Aggarwal, 2014).

2.4.3. Funcionamiento general de un clasificador

El proceso de un clasificador para interpretar un evento puede dividirse en dos partes, como se muestra en la Figura 24. La primera consiste en el entrenamiento de las clases en el cual como ya se explicó, se toma una muestra y de esta se obtienen diversas características que pasan por un procesamiento para dar como salida, valores estándar de un evento que serán enviadas al algoritmo clasificador.

La segunda parte consiste en las pruebas de predicción del clasificador en donde se obtienen características que pueden variar con el tiempo y se las compara constantemente con las clases, dando como resultado un valor de porcentaje de similitud también denominado “etiqueta” (Aggarwal, 2014).

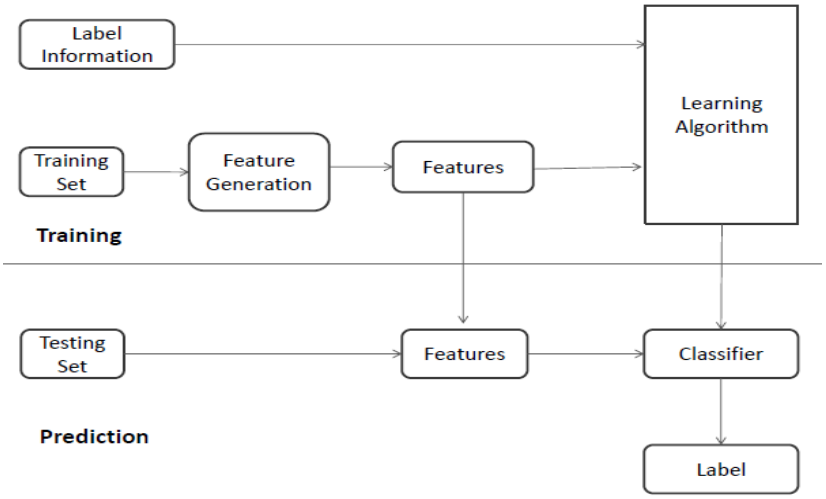


Figura 24. Esquema del proceso general de un algoritmo clasificador.

Tomado de (Aggarwal, 2014).

En la actualidad, existe una gran variedad de clasificadores los cuales varían ya sea por el tipo de características que necesitan, o por su matemática empleada.

En el presente trabajo de titulación se utilizará el algoritmo Naïve Bayes, el cual emplea matemáticas estadísticas para la extracción de características y que

además, según otros trabajos realizados, ofrece un alto porcentaje de fiabilidad y velocidad en su predicción cuando existen grandes grupos de datos (Morales & Pozo, 2017).

2.5. Clasificador Naïve Bayes

También conocido como “Clasificador bayesiano ingenuo”, es un algoritmo de tipo probabilístico, el cual asume que todas sus entradas contribuyen a la clasificación y se correlacionan mutuamente.

En otras palabras, su descripción consiste en que todas las características que ingresan en él no están relacionadas entre sí, pero que aportan de igual forma para el reconocimiento o predicción de una clase propuesta (Jadhav & Channe, 2016).

Esto puede ser considerado como un beneficio ya que se puede tener resultados favorables aun cuando existan casos donde ciertas características no se ingresen en el algoritmo.

2.5.1. Características utilizadas por el clasificador

Como se mencionó anteriormente, todo clasificador requiere de características para su funcionamiento las cuales pueden calcularse con matemáticas estadísticas, espaciales, etc.

A continuación, se expone sobre aquellas que se utilizarán con el clasificador Naïve Bayes.

2.5.1.1. Valor medio absoluto (MAV)

Corresponde al promedio obtenido por medio de la suma de los valores absolutos de una muestra y su posterior división para el número total de elementos, como se instancia en la Ecuación 1 (Morales & Pozo, 2017).

$$MAV = \frac{1}{n}\sum_{i}^{n}(|x_i|) \qquad \text{(Ecuación 1)}$$

Donde:

MAV: Valor medio absoluto.

n: Número total de elementos.

$|x_i|$: Dato absoluto.

2.5.1.2. Valor cuadrático medio (RMS)

El valor cuadrático medio, al igual que el MAV, se lo puede calcular por la sumatoria de los datos elevados al cuadrado de una muestra y su posterior división para el número total de elementos, para finalmente obtener la raíz cuadrada del resultado.

Se puede concluir que esta característica es una variación de la ecuación del valor absoluto medio, por lo que posee las mismas variables, como se observa en la Ecuación 2 (Morales & Pozo, 2017).

$$RMS = \sqrt{\frac{1}{n}\sum_{i}^{n}(x_i)^2} \qquad \text{(Ecuación 2)}$$

2.5.1.3. Varianza (VAR)

Se refiere a la sumatoria del cuadrado de las diferencias entre un dato y su valor medio del grupo al que pertenece. Para finalmente ser dividido para el número de total de elementos, como se muestra en la Ecuación 3 (Morales & Pozo, 2017).

$$VAR = \frac{1}{n}\sum_{i}^{n}(x_i - \bar{x})^2 \qquad \text{(Ecuación 3)}$$

Donde:

VAR: Varianza.

n: Número total de elementos.

x_i: Dato.

$\bar{x}$: Valor medio del grupo de datos.

2.5.1.4. Desviación estándar (STD)

La desviación estándar es un valor estadístico que se lo puede obtener al calcular la raíz cuadrada de la varianza, como se muestra en la Ecuación 4 (Morales & Pozo, 2017).

Las variables son las mismas que para calcular la varianza.

$$STD = \sqrt{\frac{1}{n}\sum_{i}^{n}(x_i - \bar{x})^2} \qquad \text{(Ecuación 4)}$$

2.5.2. Ecuaciones de Naïve Bayes

Es importante mencionar que este clasificador nace a partir del teorema de Bayes, el cual se expone en la Ecuación 5 (Morales & Pozo, 2017).

$$P(C_j/X) = \frac{P(X/C_j) * P(C_j)}{P(X)} \qquad \text{(Ecuación 5)}$$

Donde:

$P(C_j / X)$: Probabilidad posterior de una clase.

$P(X / C_j)$: Probabilidad condicional acumulada.

$P(C_j)$: Probabilidad previa de una clase.

$P(X)$: Evidencia.

$P(X / C_j)$ puede ser observada en la Ecuación 6, la cual consiste en la multiplicación continua de probabilidades condicionales cuya fórmula puede ser apreciada en la Ecuación 7 (Morales & Pozo, 2017).

$$P(X/C_j) = \prod_{i=1}^{n} P(x_i/C_j) \qquad \text{(Ecuación 6)}$$

Donde:

$P(X / C_j)$: Probabilidad condicional acumulada.

X: Vector de características de una clase j.

$P(x_i / C_j)$: Probabilidad condicional de i características con la clase j.

$$P(x_i/C_j) = \frac{1}{\sqrt{2\pi\delta_{ic}^2}} * e^{-\frac{(x_i-u_{ic})^2}{2\delta_{ic}^2}} \qquad \text{(Ecuación 7)}$$

Donde:

$P(X_i / C_j)$: Probabilidad condicional.

x_i: Características del vector X.

C_j: Clase a ser analizada.

δ_{ic}: Desviación estándar de la característica de entrenamiento i de la clase j.

u_{ic}: Valor medio de la característica de entrenamiento i de la clase j.

$P(C_j)$ es un valor que varía según las clases que se tenga entrenadas. En este proyecto de titulación solo se entrenará las clases abierto y cerrado, por lo cual el valor de esta constante será 0,5 debido a que hay 50% de probabilidad que sea abierto y 50% que sea cerrado, como se muestra en la Ecuación 8 (Morales & Pozo, 2017).

$$P(C_j) = 0{,}5 \qquad \text{(Ecuación 8)}$$

P(X) consiste en la sumatoria de los productos entre la probabilidad condicional acumulada y la probabilidad previa de las clases abierto y cerrado. Esto se entiende de mejor forma en la Ecuación 9 (Morales & Pozo, 2017).

$$P(X) = \sum_{j=1}^{m} \left(\prod_{i=1}^{n} \left(P\left(x_i / C_j\right) \right) * P\left(C_j\right) \right) \qquad \text{(Ecuación 9)}$$

Donde:

P(X): Evidencia.

$P(x_i/C_j)$: Probabilidad condicional acumulada de la clase i.

$P(C_j)$: Probabilidad previa de la clase i.

2.6. Elementos del sistema microcontrolado

El sistema microcontrolado planteado para el presente proyecto de titulación puede dividirse en dos partes, la primera corresponde a la captura de datos electromiográficos y la segunda al procesamiento de las señales y control de la mano robótica. A continuación, se detalla una breve descripción de los componentes involucrados.

2.6.1. DSD TECH HM – 11

Es un módulo de conexión Bluetooth el cual ha sido elegido para formar parte del sistema microcontrolado debido a que utiliza conexión 4.0 de esta tecnología, misma que es utilizada por el sensor MYO Armband para transmitir la información, se encuentra disponible para la venta a un precio de 9,99 USD, y se lo puede apreciar en la Figura 25 (DSD TECH, 2018).

Posee las mismas características que el módulo HM – 10 solo variando en el núcleo. Algunas especificaciones especiales se muestran a continuación:

- Su procesador es un CC2541F256, que permite conectividad con dispositivos Apple desde iOS 7 y Android 4.3 en adelante.
- Posee seis pines de conexión, los cuales son: VCC, GND, RX, TX, STATE y EN.
- Es necesario que sea alimentado con voltaje desde 3,6 V hasta 6 V.
- Su velocidad de transmisión por defecto es 9600 baudios, pero se la puede aumentar hasta 115200 baudios (DSD TECH, 2018).

Figura 25. DSD TECH HM – 11.

Tomado de (DSD TECH, 2018).

2.6.2. Arduino Nano

Es una placa la cual tiene como microcontrolador un chip ATmega328P, ofreciendo las misas funcionalidades que una placa Arduino Uno, por un precio de 22 USD

(Arduino, 2019a). Se puede programar en el directamente desde el IDE de Arduino y se lo escogió para este trabajo de titulación debido a su tamaño reducido y a su compatibilidad con la librería necesaria para utilizar el módulo HM – 11 con el MYO Armband. A continuación, se presentan algunas características importantes:

- Su voltaje de operación es de 5 V, el cual puede ser suministrado por su puerto USB – Mini B.
- Posee una velocidad de reloj de 16 MHz.
- Puede soportar un voltaje de entrada de 7 V a 12 V.
- Tiene un total de 22 pines digitales, de los cuales 6 son PWM y 2 son de comunicación serial, mismos que se pueden observar en la Figura 26 (Arduino, 2019a).

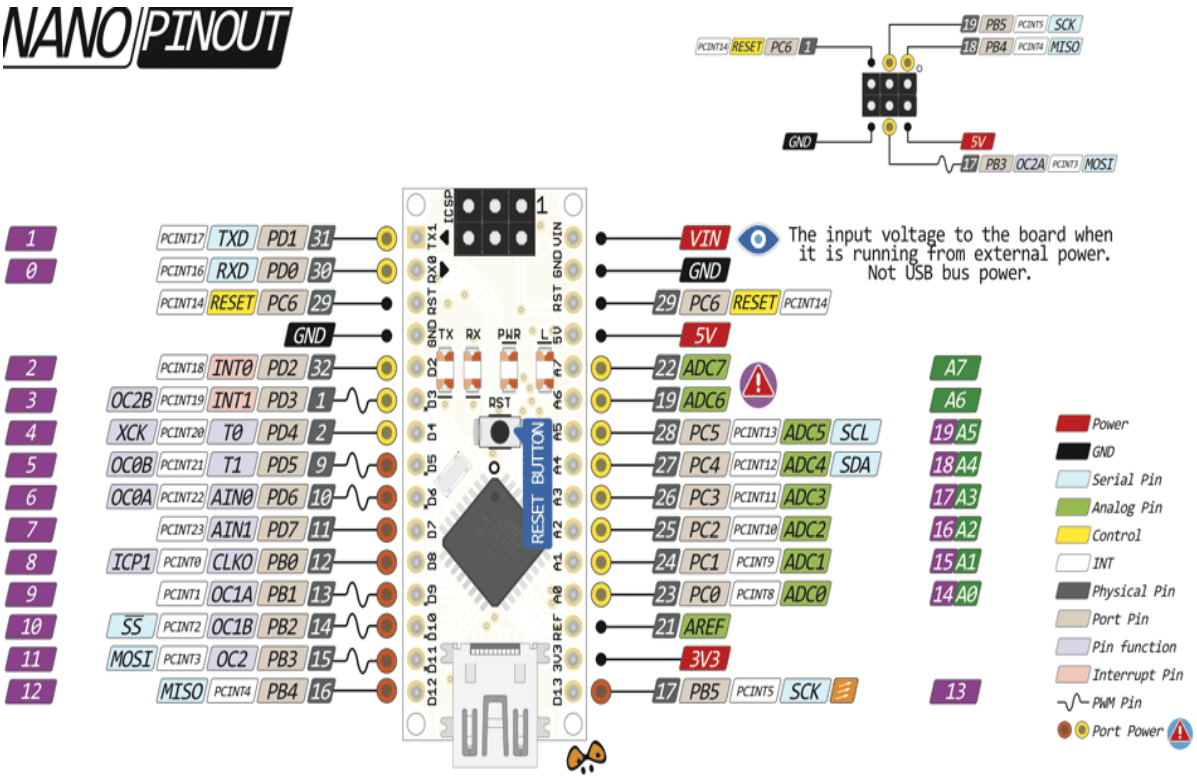


Figura 26. Arduino Nano Pinout.

Tomado de (101 Components, 2018).

2.6.3. Microcontrolador STM32F103C8

El microcontrolador STM32F103C8, también denominado “Blue Pill” es una placa la cual tiene como núcleo, un procesador ARM Cortex – M3 con una arquitectura de 32 bits que la hace superior en velocidad de procesamiento respecto a otras placas de 16 bits, y tiene un costo económico de 18,99 USD (Keil, 2019). Fue escogido para este proyecto de titulación debido a tu tamaño reducido y su velocidad de procesamiento eficiente ya que se tiene gran cantidad de información que debe ser procesada rápidamente para evitar colapsos o grandes retardos en tiempos de respuesta.

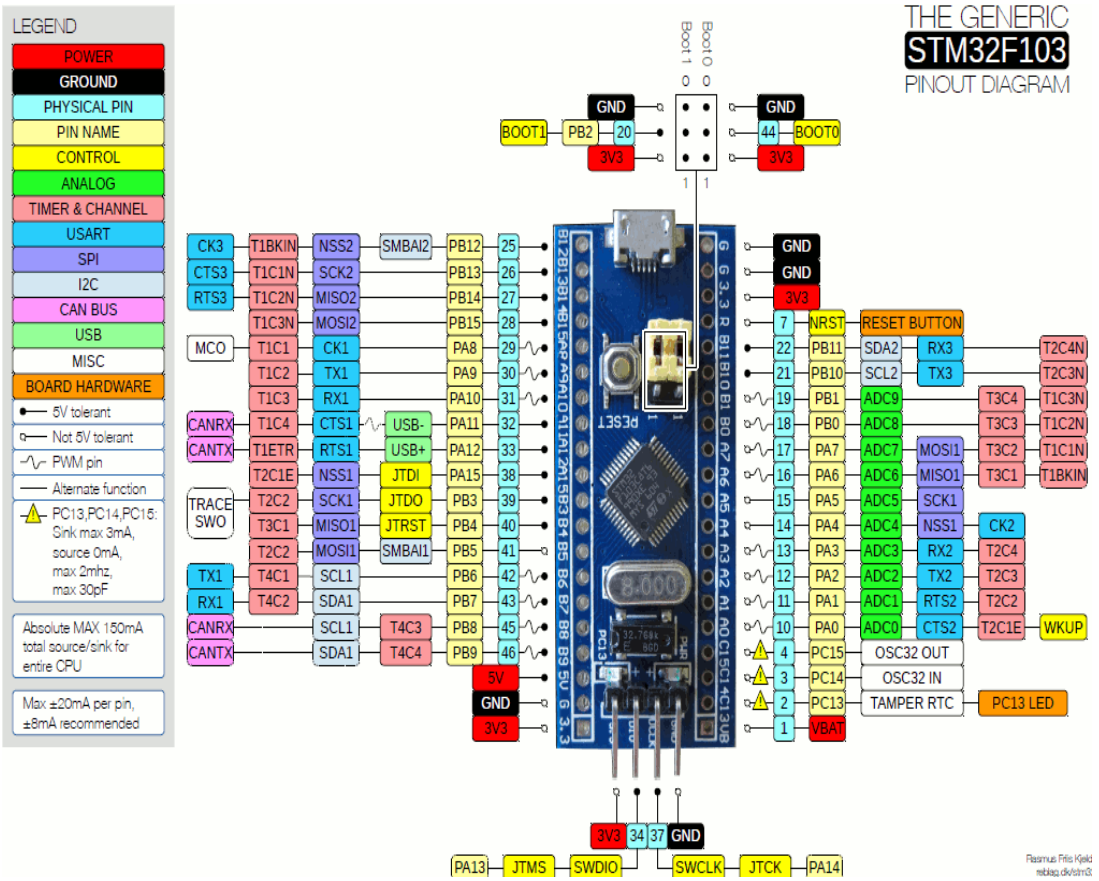


Figura 27. STM32F103C8 Pinout.

Tomado de (Stm32duino, 2019).

A continuación, se describen algunas características relevantes:

- Tiene una memoria ROM de 64 Kb, mientras que su RAM es de 20 Kb.
- Posee una velocidad de reloj de 72 MHz.
- Admite comunicación SPI, I2C, CAN, USART y USB como se puede observar en la Figura 27.
- Para su programación es necesario utilizar un módulo auxiliar TTL.
- Refiriéndose al Software para desarrollo, se recomienda usar Keil uVision junto con CubeMX, programas especializados en procesadores de 32 bits (Keil, 2019).

2.6.4. Componentes auxiliares

Módulo USB a TTL CP2102: Puede observarse en la Figura 28 y permite configurar y programar el microcontrolador STM32 al ser conectado en alguno de sus puertos de comunicación serial TX y RX.

Convertidor Voltaje DC Step-Down MP1584EN: Se lo puede apreciar en la Figura 29 y permite regular un voltaje de salida constante de 5 V a 2 A. Con el objetivo de que no exista energía variable que afecte el funcionamiento del sistema o lo dañe.

Módulo Bluetooth HC – 06: Propuesto para utilizarse en el STM32 con el objetivo de observar la información procesada de forma inalámbrica. Puede observarse en la Figura 30.

Figura 28. Módulo USB a TTL CP2102.

Tomado de (Deal Extreme, 2019).

Figura 29. Convertidor Step-Down MP1584EN.

Tomado de (Naylamp Mechatronics, 2019).

Figura 30. Módulo Bluetooth HC – 06.

Tomado de (MovilTronics, 2019).

Una vez explicadas las partes involucradas tanto en el sistema microcontrolado, como en la mano robótica a utilizar y entendida la teoría relacionada con algoritmos clasificadores, señales electromiográficas y el sensor MYO Armband.

Se procede a la aplicación de estos conocimientos en el desarrollo e implementación del control de la mano robótica por medio de señales electromiográficas.

3. DESARROLLO E IMPLEMENTACIÓN

En el presente capítulo, se expone de forma completa y detallada el proceso de desarrollo del sistema microcontrolado y la captura de las ocho señales electromiográficas proporcionadas por el sensor MYO Armband. Además, se presenta la implementación del algoritmo clasificador Naïve Bayes para la identificación del movimiento abierto y cerrado de la mano humana para el posterior control de la mano robótica.

3.1. Esquema general del sistema de control

De manera general, el sistema de control propuesto en este trabajo de titulación puede ser resumido en el esquema de la Figura 31, donde se explica paso a paso la función que cumple cada componente para lograr el movimiento de la mano robótica.

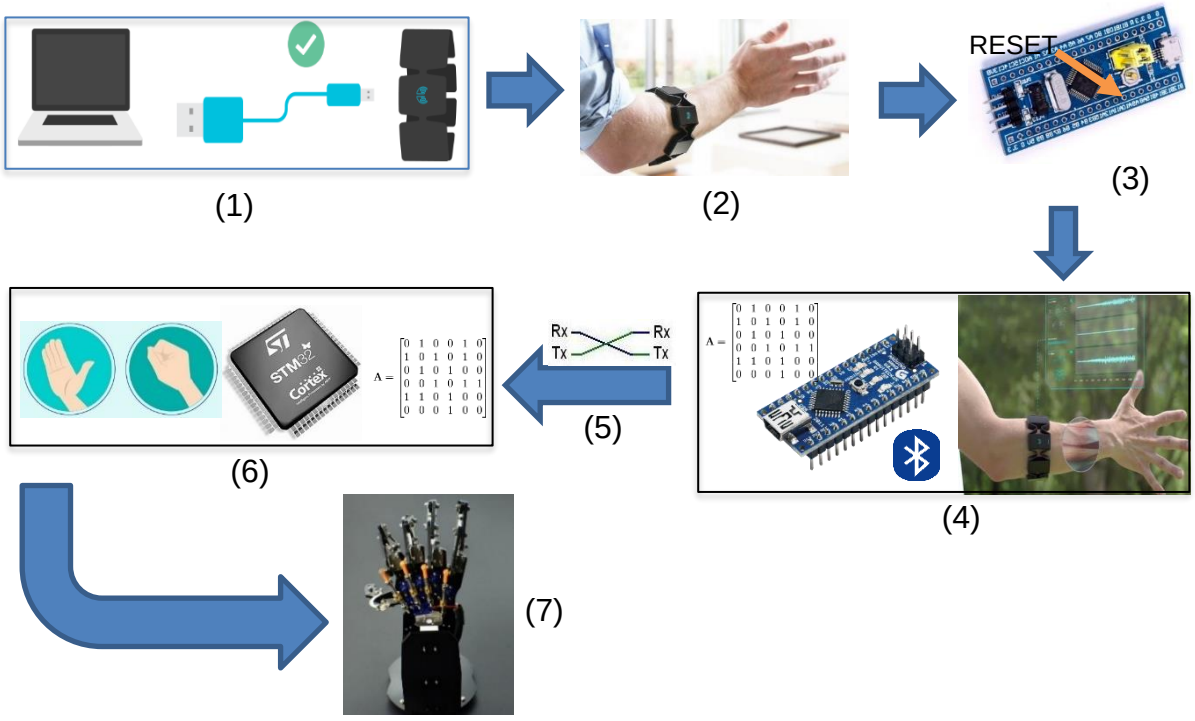


Figura 31. Esquema general del sistema de control de la mano robótica.

1. Se enciende el sensor MYO Armband conectándolo por 3 segundos al computador desde el puerto USB.
2. Se coloca el dispositivo en la parte superior del antebrazo derecho que, al sincronizarse con el módulo HM – 11, emite una vibración al usuario mientras que el LED se mantiene de color azul.
3. El STM32 es considerado como el corazón del sistema ya que en él se puede modificar la cantidad de datos a trabajar, que son tomados en una matriz de

n filas por ocho columnas. Además, se aplica el clasificador y se generan las señales PWM que controlan la mano robótica.

Continuando con los pasos, se presiona el botón RESET del STM32 para enviar por comunicación serial un byte de inicio con el valor correspondiente a las filas requeridas hacia el ATmega328P, iniciando el proceso.

4. ATmega328P recibe dicho byte y abre la conexión con el sensor que comienza a transmitir datos EMG hacia el módulo HM – 11, construyendo en el microcontrolador la matriz solicitada.
5. Una vez terminada de construir la matriz de datos, el microcontrolador ATmega328P los envía por comunicación serial hacia el STM32.
6. El microcontrolador recibe dichos datos y reconstruye la matriz de forma idéntica.

   Posteriormente, se obtienen las características de la señal, se aplica el algoritmo Naïve Bayes y se genera un porcentaje el cual se evalúa en un margen de predicción que determina si el movimiento realizado es abierto o cerrado.
7. Según la predicción, el STM32 genera las señales PWM para el control de los servomotores de la mano robótica, logrando así que esta abra o cierre sus dedos.

Para una mejor comprensión, se puede observar con una perspectiva distinta el esquema y los pasos presentados anteriormente en el diagrama de flujo expuesto en la Figura 32.

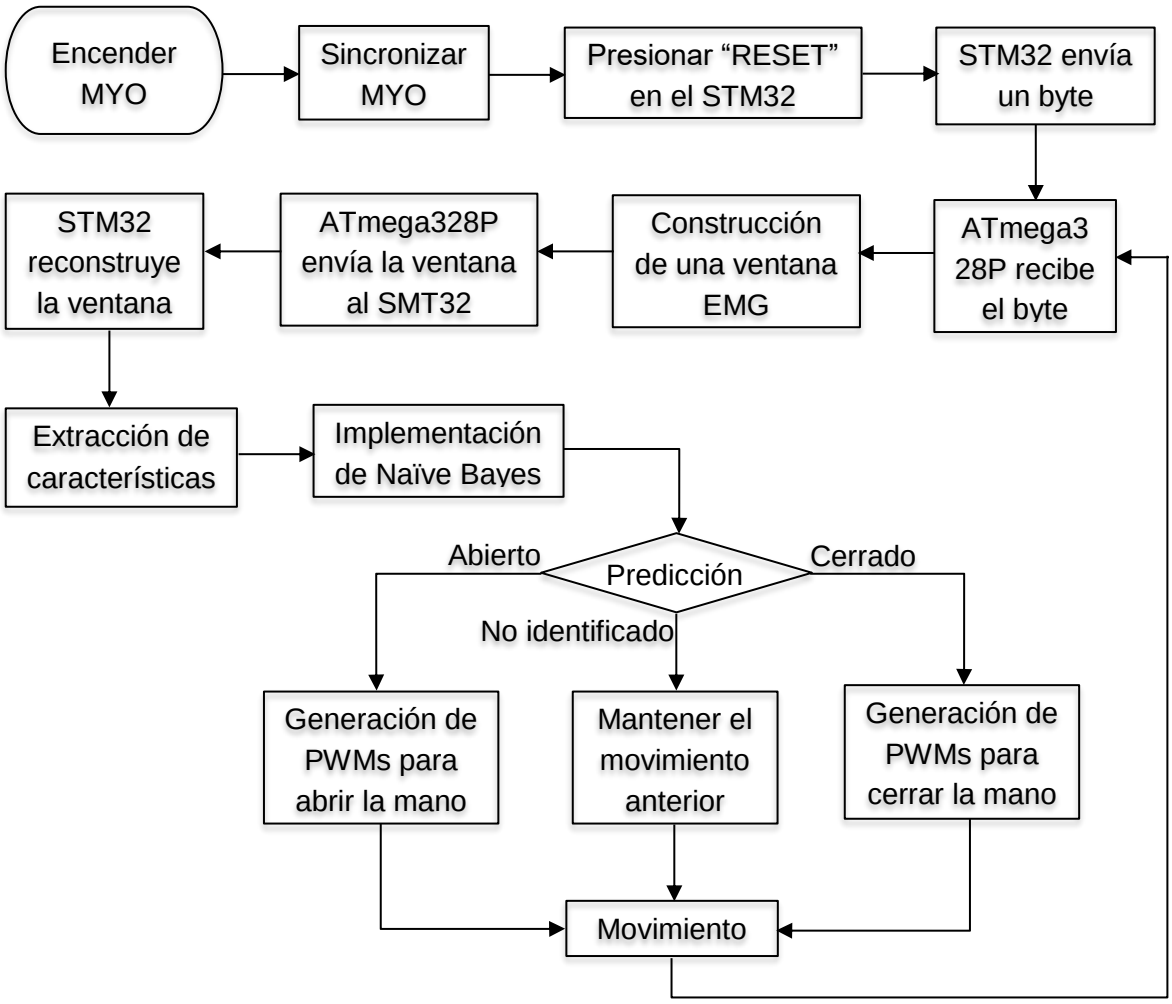


Figura 32. Diagrama de flujo del funcionamiento del sistema de control.

Una vez comprendido el esquema sistema, se explica detalladamente el proceso de desarrollo de este, que permitirá alcanzar el objetivo.

3.2. Adaptación del módulo Bluetooth HM – 11

Para adaptar el módulo HM – 11 al sistema microcontrolado, se procede a cambiar su versión de Firmware con una que se pueda conectar y comunicar con el sensor MYO Armband, mediante una librería llamada "MyoBridge" creada por Valentín Roland y adecuando los pasos presentados por Ahmed Alsharif en Raquena Engineering (Alsharif, 2017).

Para ello se requiere el apoyo de un microcontrolador ATmega328P que se encuentre en la placa de desarrollo Arduino UNO. Dicho esto, se comienza soldando tres cables en los pines que hacen referencia a los registros “DATA”, “CLOCK” y “RESET”, que se pueden observar en la Figura 33.

Después, se procede a descargar la carpeta de la aplicación CCLoader que se la puede encontrar en GitHub y que al instalarse, sirve para programar directamente en el procesador (CC2541) del HM – 11 con fines de cambio de Firmware (RedBearLab, 2017). Igualmente desde el repositorio de GitHub, se descarga la carpeta “MyoBridge-master” (Roland, 2017).

Una vez se tenga estos elementos en el computador, se abre el IDE de Arduino y se carga en la placa UNO el programa “CCLoader.ino” que se encuentra en la carpeta de la aplicación.

Posteriormente, se procede a conectar el módulo MH – 11 con la placa Arduino tal y como se muestra en la Figura 34. A continuación, se abre la línea de comandos (CMD) del computador, se apunta hacia la dirección donde se encuentra la carpeta CCLoader y se ejecuta la aplicación mediante la siguiente línea de comando:

“CCLoader.exe <COM> <Archivo BIN> <Dispositivo>”

Donde:

- **COM:** Número del puerto al que está conectado el Arduino UNO.
- **Archivo BIN:** Dirección del archivo BIN en la carpeta MyoBridge-master.
- **Dispositivo:** 0 para Arduino UNO, 1 para cualquier otro.

Ejemplo: C:\Users\JD\Downloads\CCLoader\Windows> CCLoader.exe 8 C:\Users\JD\Downloads\MyoBridge-master\myobridge_firmware\Hex\MyoBridge_CC2541.bin 0

Una vez ejecutada esta línea de comando, el Arduino UNO actualiza el Firmware del HM – 11, para que sea compatible con la librería MYO Bridge en el microcontrolador ATmega328P y así establecer una comunicación exitosa con el MYO Armband para poder solicitar las señales electromiográficas.

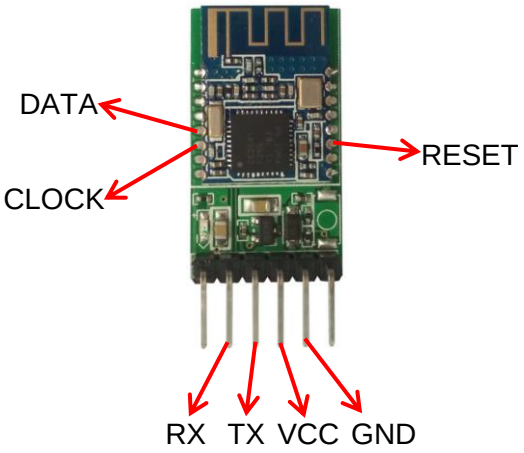


Figura 33. Pines a soldar en el módulo HM – 11.

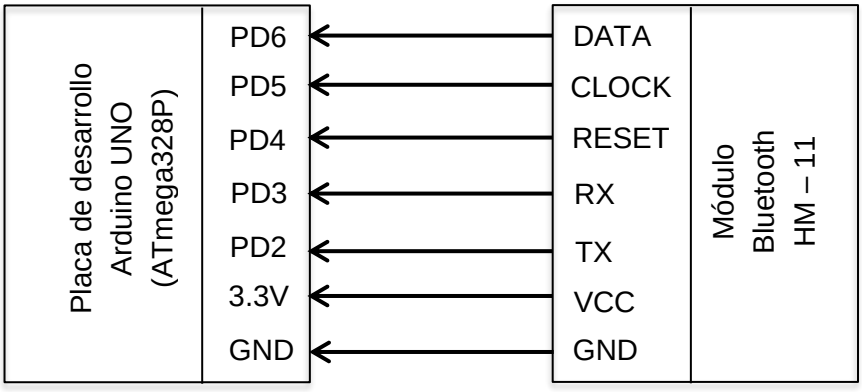


Figura 34. Conexión con el Arduino Uno.

3.3. Configuración del ATmega328P y obtención de señales EMG

Posterior a la actualización del Firmware del módulo HM – 11 y continuando con la adaptación de los pasos presentados en Raquena Engineering (Alsharif, 2017), se procede a conectar los componentes como se explica en la Figura 35.

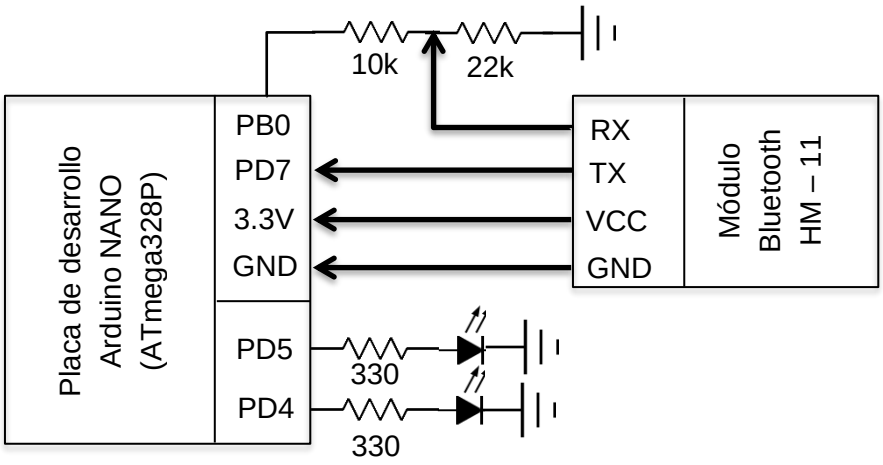


Figura 35. Conexión del módulo HM – 11 con el Arduino Nano.

Es importante mencionar que se debe realizar un divisor de voltaje como protección entre la conexión de TX del Arduino Nano y RX del HM – 11 debido a que la señal que la placa entrega es de 5V, mientras que el módulo solamente soporta hasta 3.3V.

Para ello se debe colocar dos resistencias en serie, una de 10kΩ y otra de 22kΩ que estará conectada a una tierra común. Entre estas resistencias se conecta un cable que va hacia el pin RX del módulo. Así mismo, se conecta dos LEDs con a una resistencia de 330Ω a manera de indicadores de estado.

Completada esta conexión, se procede conectar la placa Nano al computador y se agrega la librería MyoBridge al IDE de Arduino colocando la carpeta de esta en el directorio “Libraries” del programa. Realizado esto, se puede utilizarla con solo incluirla en el proyecto.

Es importante resaltar que esta librería incluye un ejemplo de su operación (que se lo puede encontrar en el enlace de GitHub), mismo que se tomará como punto de partida para lograr el objetivo de este trabajo de titulación (Roland, 2017).

Dicho esto, en seguida se presentan una serie de diagramas de flujo representativos del código programado para obtener las señales EMG del sensor y enviarlas al STM32, el cual involucra las siguientes librerías:

- **MyoBridge:** Utilizada para establecer comunicación con el MYO Armband.
- **SoftwareSerial:** Sirve para configurar cualquier puerto de la placa Nano como puerto de comunicación serial.
- **TimerOne:** Es necesaria para trabajar con interrupciones en el microcontrolador y así tener una mejor sincronización del sistema.

3.3.1. Programa Principal

Corresponde a la estructura primordial del programa que se lo puede encontrar en la Figura 36.

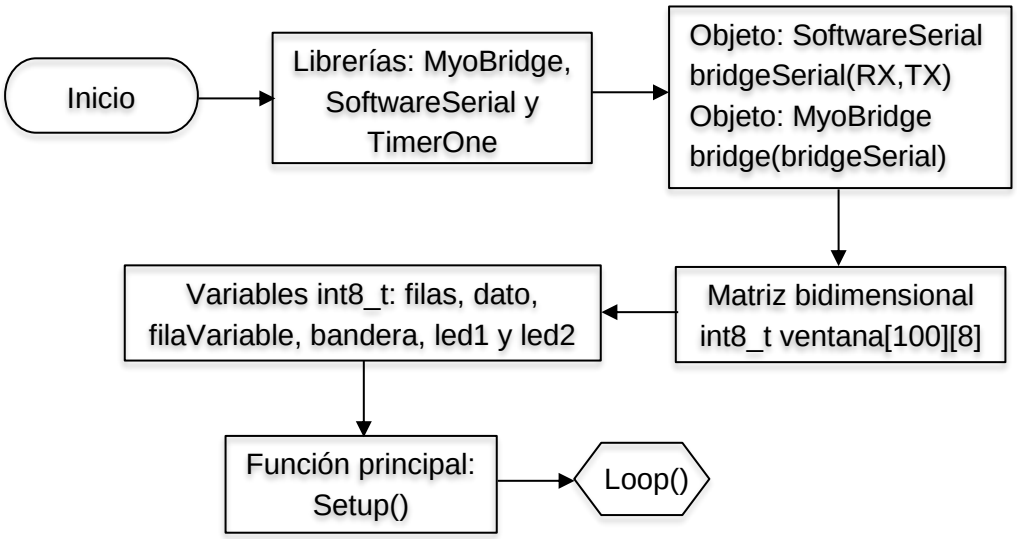


Figura 36. Diagrama de flujo general del programa.

Como se puede observar anteriormente, en este diagrama se encuentra tanto la inclusión de librerías, como instanciación de objetos y declaración de variables que

serán utilizadas a lo largo del código. A continuación, se explica brevemente sobre la siguiente función relevante incluida en el diagrama de flujo.

"SoftwareSerial bridgeSerial(RX,TX)"

Donde:

- **SoftwareSerial:** Librería para configurar puertos como seriales (Arduino, 2019f).
- **bridgeSerial:** Nombre del objeto.
- **RX,TX:** Pines del Arduino Nano a configurar.

3.3.2. Función: Setup

Es la función que se ejecuta primero al iniciar el programa y se la puede observar en el diagrama de flujo expuesto en la Figura 37. En esta función se tiene la inicialización de las comunicaciones así como configuración de la interrupción a utilizarse y los LEDs indicadores. A continuación, se explica brevemente sobre algunas funciones incluidas en el diagrama de flujo mostrado.

"Serial.begin(115200)"

"bridgeSerial.begin(115200)"

"bridge.begin(printConnectionStatus)"

Donde:

- **Serial:** Puerto serial del Arduino Nano (Arduino, 2019e).
- **bridgeSerial:** Objeto de la clase SoftwareSerial.
- **bridge:** Objeto de la clase MyoBridge.
- **begin:** Método para la inicialización del enlace (Arduino, 2019e).
- **115200:** Velocidad de comunicación en baudios (Arduino, 2019c).

- **printConnectionStatus:** Método que informa sobre el estado actual del enlace entre el sensor y el sistema.

"Timer1.attachInterrupt(cInterrupcion)"

Donde:

- **Timer1:** Temporizador del Arduino, de la librería TimerOne.
- **attachInterrupt:** Método para habilitar una interrupción (Prometec, 2019).
- **cInterrupcion:** Nombre de la función que se realizará cuando se detecte la interrupción.

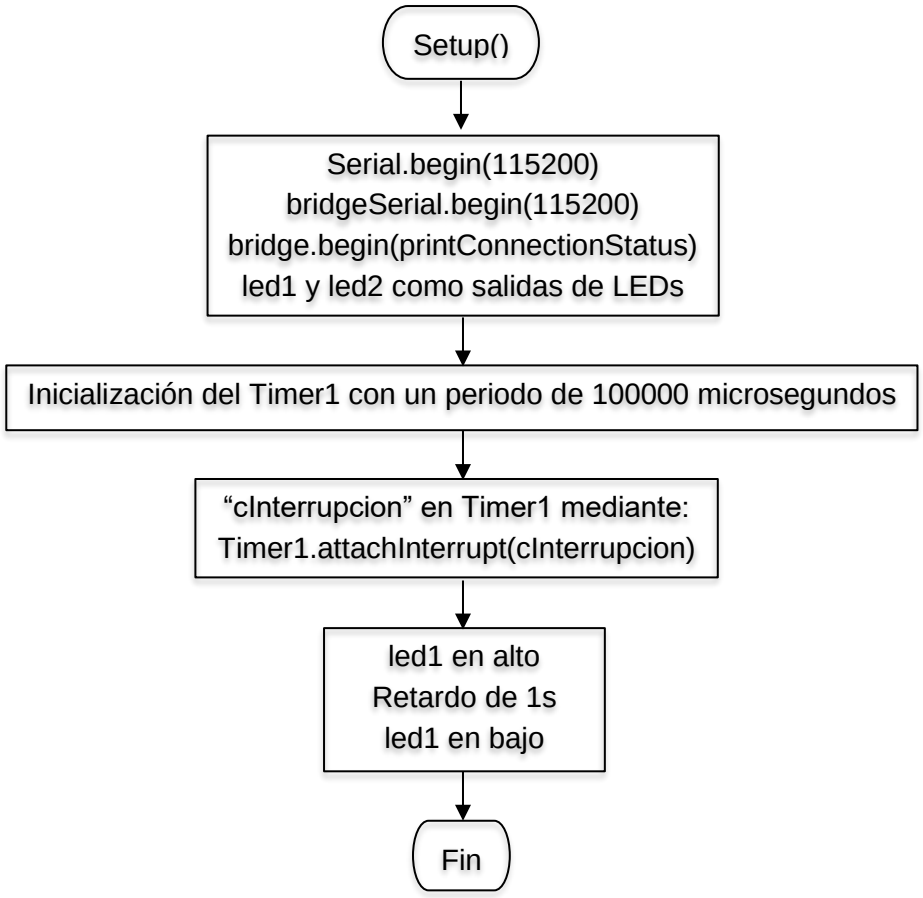


Figura 37. Función principal Setup.

3.3.3. Función: Loop

Es una función cuyo objetivo es obtener las señales EMG del sensor de forma constante mediante la función “getEMGData”. Un diagrama de flujo representativo de la función se lo puede encontrar en la Figura 38.

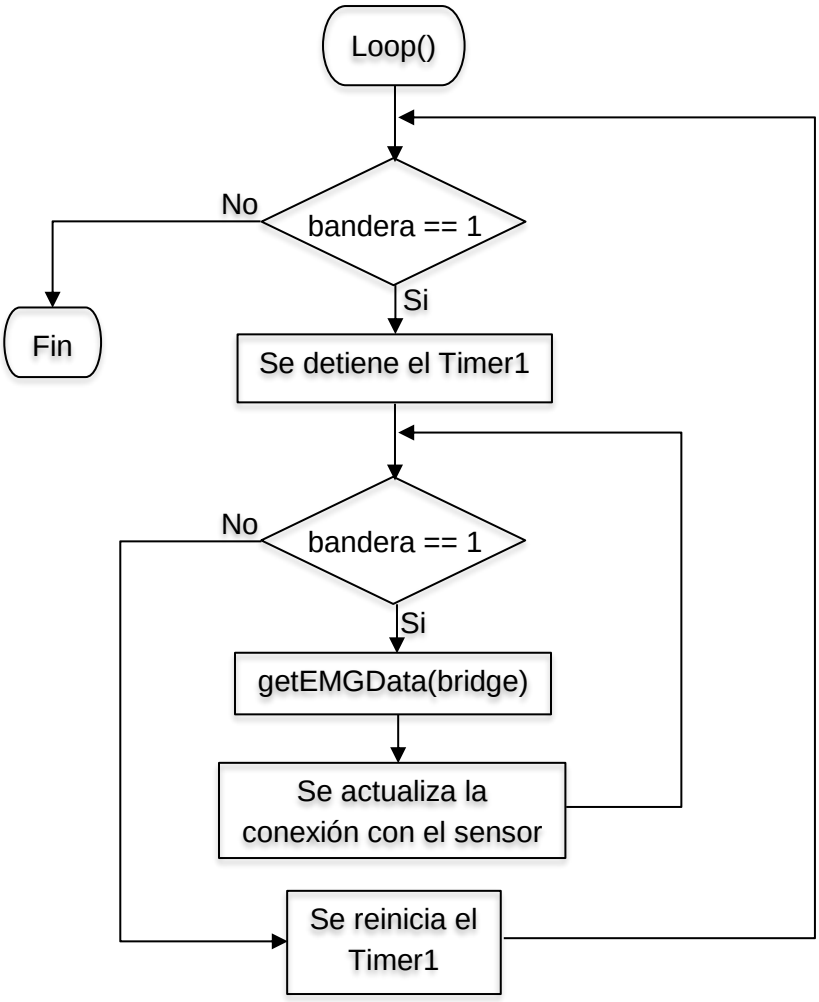


Figura 38. Función Loop.

3.3.4. Función: printConnectionStatus

Esta función de la puede encontrar en la función principal Setup y como se mencionó anteriormente, sirve para informar sobre el estado actual de la conexión del MYO Armband y el módulo HM – 11.

Tiene como parámetro de entrada una variable “status” de tipo “MyoConnectionStatus” y mediante el siguiente código se imprime en pantalla:

“Serial.println(bridge.connectionStatusToString(status))”

Donde:

- **Serial:** Puerto serial del Arduino Nano.
- **println:** Método para imprimir una línea en el puerto Serial.
- **bridge:** Objeto de la clase MyoBridge.
- **connectionStatusToString:** Método que obtiene el estado del sensor con respecto a la comunicación Bluetooth (MyoBridge Library, 2019b).
- **status:** Variable que indica el estado del enlace con los valores expuestos en la Tabla 2 (MyoBridge Firmware, 2019).

Tabla 2.

Valores de la variable status.

Valor	Descripción
MYB_STATUS_UNKOWN	Estado desconocido, debido a un error.
MYB_STATUS_INIT	El sistema se está inicializando.
MYB_STATUS_SCANNING	El sistema está escaneando dispositivos MYO Armband cercanos.
MYB_STATUS_CONNECTING	El sistema se está conectando al MYO Armband.
MYB_STATUS_DISCOVERING	El sistema está descubriendo la configuración actual del MYO Armband.
MYB_STATUS_BUSY	El sistema está ocupado con otra conexión.
MYB_STATUS_CONNECTED	El sistema se encuentra conectado.

Adaptado de (MyoBridge Firmware, 2019).

3.3.5. Función: showEMGData

Función que sirve para almacenar los valores EMG obtenidos por el sensor en un arreglo bidimensional llamado “ventana”, que puede ser apreciada en el diagrama de flujo de la Figura 39.

Tiene como parámetro de entrada un arreglo de tipo int8_t nombrado “EMGData” con una capacidad para 8 elementos.

Esto se debe a que los valores EMG entregados por el MYO Armband llegan de forma secuencial desde el sensor 1 hasta el sensor 8, lo cual implica que MyoBridge deba guardarlos en un arreglo para que no existan confusión de datos. En otras palabras, al guardar los valores electromiográficos en EMGData se tiene una línea ordenada de señales.

Sin embargo, el sistema controlado requiere de más datos para su funcionamiento. Por lo que, posterior a la recolección de una línea EMG, los valores se guardan en el arreglo “ventana” para almacenar un número alto de filas que juntas forman una colección de señales EMG que posteriormente será enviada hacía el microcontrolador STM32.

A continuación, se explica brevemente sobre la siguiente función relevante incluida en el diagrama de flujo.

“ventana[filas][columna] = EMGData[columna]”

Donde:

- **ventana:** Arreglo bidimensional de 100 filas por 8 columnas.
- **filas:** Número de fila en el arreglo.
- **columna:** Número de columna en el arreglo.
- **EMGData:** Arreglo unidimensional de 8 elementos.

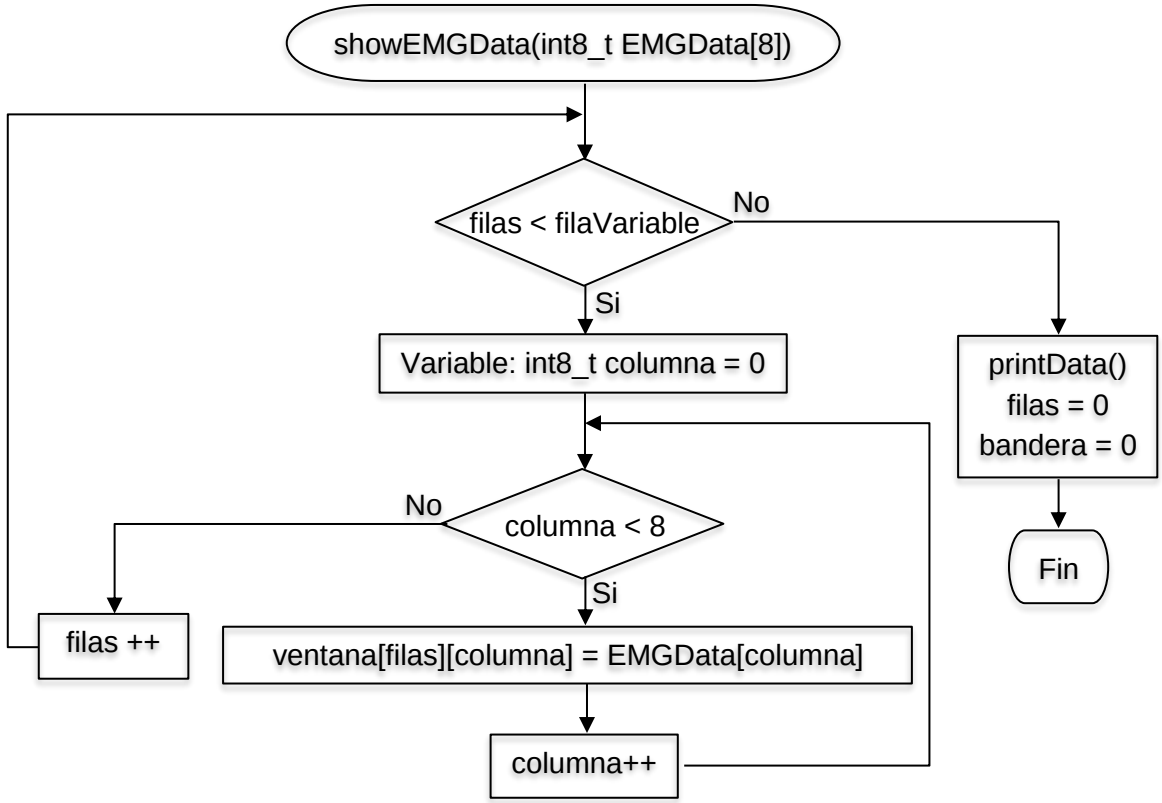


Figura 39. Función showEMGData.

3.3.6. Función: getEMGData

Esta función puede considerarse como la fundamental para recolectar las señales electromiográficas ya que posee los métodos necesarios de la librería MyoBridge para adquirir los valores que posteriormente serán almacenados. Razón por la cual, tiene como parámetro de entrada el objeto MyoBridge declarado al inicio.

Un diagrama de flujo que describe la estructura de esta función se la puede encontrar en la Figura 40 y en seguida, se procede a explicar sobre las funciones implicadas para una mejor comprensión.

“bridge.setEMGDataCallBack(showEMGData)”

Donde:

- **bridge:** Objeto de la clase MyoBridge.
- **setEMGDataCallBack:** Método que captura las señales y las guarda (MyoBridge Library, 2019a).
- **showEMGData:** Función en donde se graba los datos para su transmisión.

"bridge.setEMGMode(EMG_MODE_SEND)"

Donde:

- **setEMGMode:** Método para establecer el modo de transmisión EMG (MyoBridge Library, 2019a).
- **EMG_MODE_SEND:** Comando para que el sensor entregue valores electromiográficos filtrados (sin ruido) (MyoBridge Library, 2019a).

"bridge.disableSleep()"

Donde:

- **disableSleep:** Método que desactiva el modo reposo del MYO Armband para que este no detenga la transmisión de valores (MyoBridge Library, 2019a).

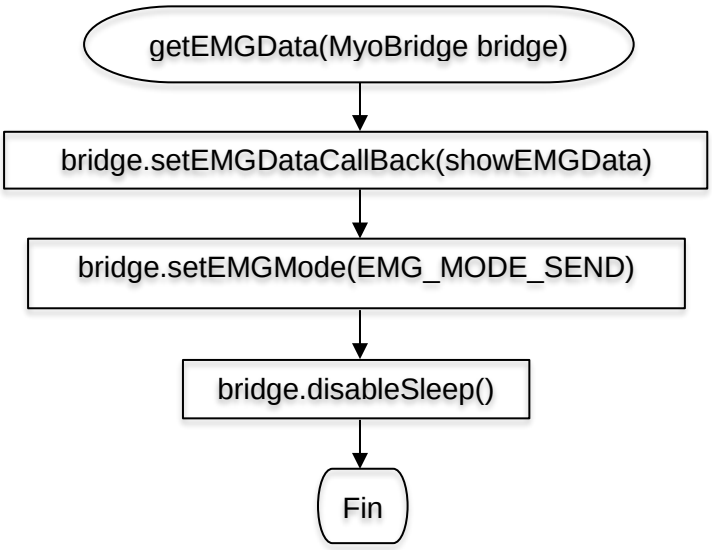


Figura 40. Función getEMGData.

3.3.7. Función: printData

PrintData es la encargada de transmitir de forma ordenada los valores electromiográficos al microcontrolador de 32 bits una vez se haya terminado de guardarlos en el arreglo “ventana” mediante la función “write”, la cual transmite la información en bytes (Arduino, 2019g).

Esto se lo realiza por comunicación serial en donde el STM32 es el maestro que solicita los datos y el Arduino Nano es el esclavo que los entrega. Una vez este termine de recibir los valores y reconstruirlos, volverá a pedir al esclavo nuevos datos.

Un diagrama de flujo de la mencionada función se la puede ver en la Figura 41.

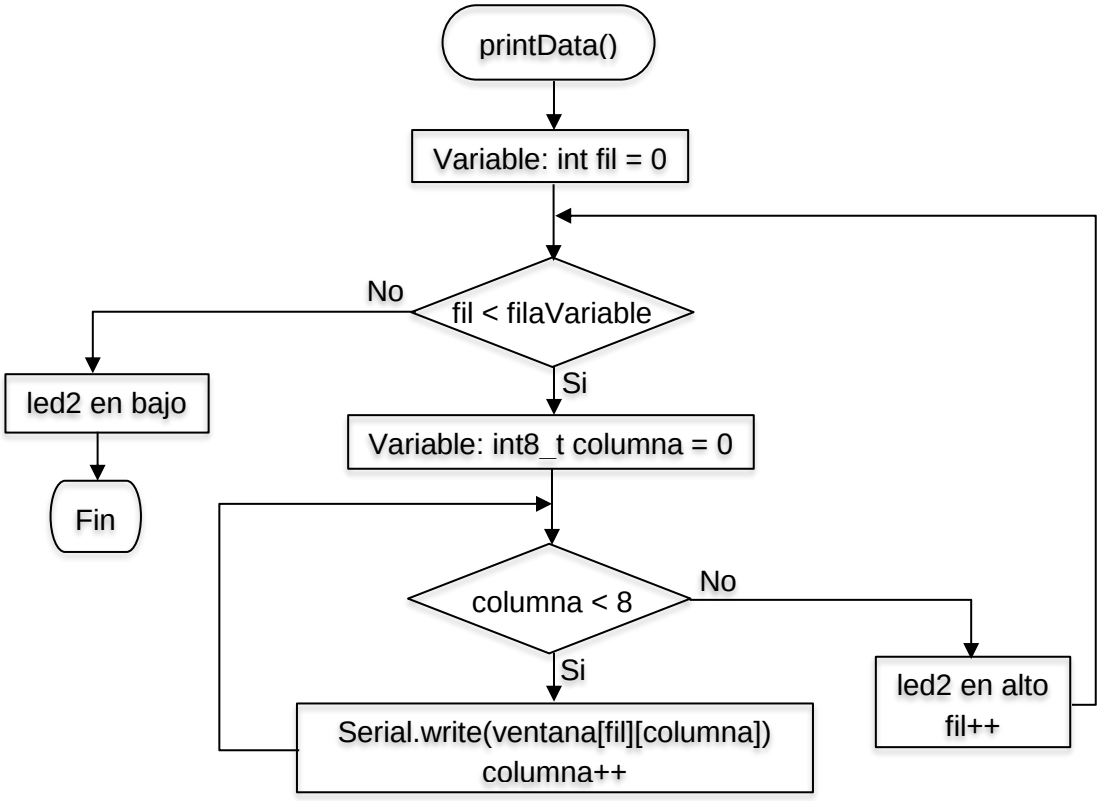


Figura 41. Función printData.

3.3.8. Interrupción por Timer1

Como se mencionó anteriormente, Timer1 es una interrupción de la librería TimerOne, la cual habilita temporizadores en el Arduino Nano (ElectroTools, 2016). Esta ejecutará periódicamente (según un tiempo establecido en microsegundos) el cuerpo de la interrupción llamado “cInterrupcion” que evalúa el puerto serial del ATmega328P en espera de que exista algún dato en el buffer, como se muestra en la Figura 42.

El STM32 envía un byte de inicio (bandera) al microcontrolador, el cual contiene la información del número de datos EMG que serán solicitados al sensor MYOArmband, dando inicio al proceso de captura y transmisión.

Posteriormente, se reinicia el Timer1 y se repite el proceso. Sus dos métodos relevantes son:

- **Serial.available():** Verifica que exista algún dato entrante en el puerto (Arduino, 2019b).
- **Serial.read():** Lee el dato recibido desde el puerto Serial (Arduino, 2019d).

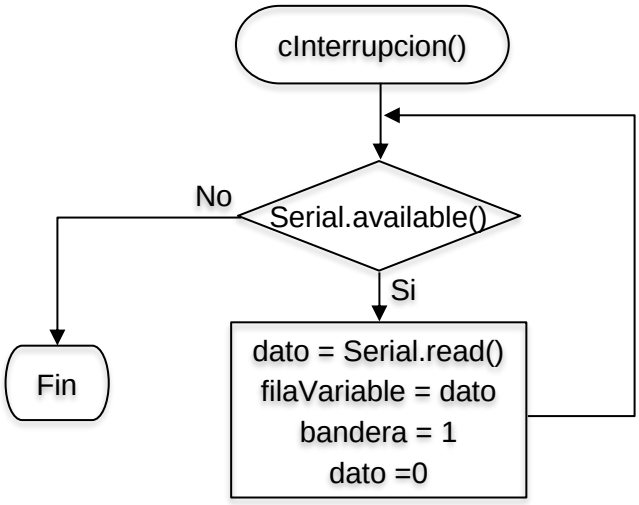


Figura 42. Función cInterrupcion.

3.4. Conexión del microcontrolador STM32F103

Antes de explicar la configuración y programación del microcontrolador de 32 bits, es importante exponer como primer punto, el diagrama de conexión entre estas placas con los elementos complementarios que son, el módulo HC – 06, los cinco servomotores genéricos de la mano robótica y el módulo USB – TTL.

El diagrama correspondiente se lo puede apreciar en la Figura 43.

Además, utilizará cuatro LEDs de alta luminosidad para propósitos informativos del estado de procesamiento y predicción así como también, la aplicación de un divisor de voltaje entre el puerto TX del STM32 y el RX del módulo HC – 06 como se realizó anteriormente.

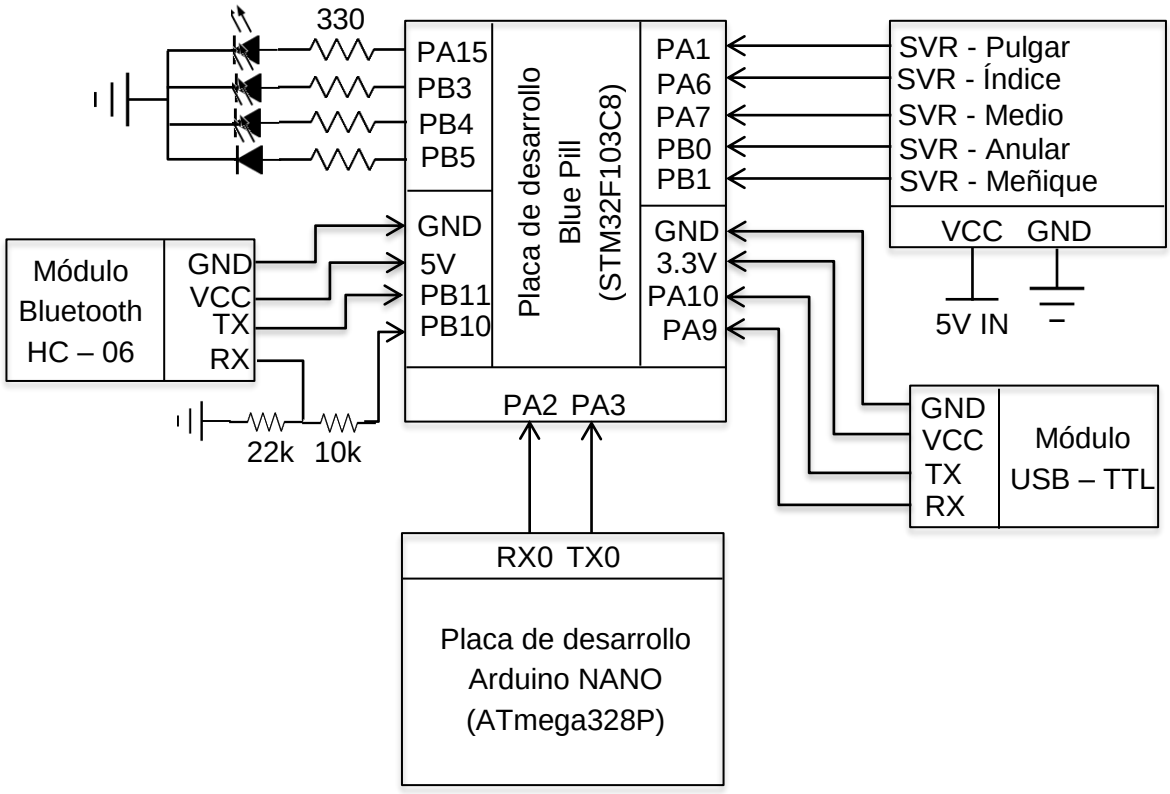


Figura 43. Conexión del STM32 con el ATmega328P y sus complementos.

3.5. Software necesario para la programación

En primer lugar se procede a utilizar el Software “STM32CubeMX” que sirve para configurar en interfaz gráfica, aspectos del procesador del STM32 como temporizadores, PWM, interrupciones, entre otros (STMicroelectronics, 2019). Es una herramienta muy útil ya que una vez configurado lo deseado, se construye un proyecto en lenguaje C traduciendo a código las configuraciones realizadas de forma gráfica.

El Software que se utilizará para editar dicho proyecto es “Keil uVision 5” que sirve para programar directamente en los procesadores ARM (ARM Keil, 2019). La aplicación que se utilizará como complemento para quemar el programa en el STM32 mediante el módulo USB – TTL es “STM32 Flash Loader Demonstrator”.

Para hacer esto, primeramente se procede a colocar el jumper del SMT32 en modo programación como se muestra en la Figura 44 y se presiona el botón RESET para entrar en él. Después, se procede a abrir STM32 Flash Loader y se selecciona el puerto en el que está conectado el módulo USB – TTL y se quema el programa. Finalmente, se coloca nuevamente el jumper en modo de operación y se presiona el botón de RESET para ejecutar el programa.

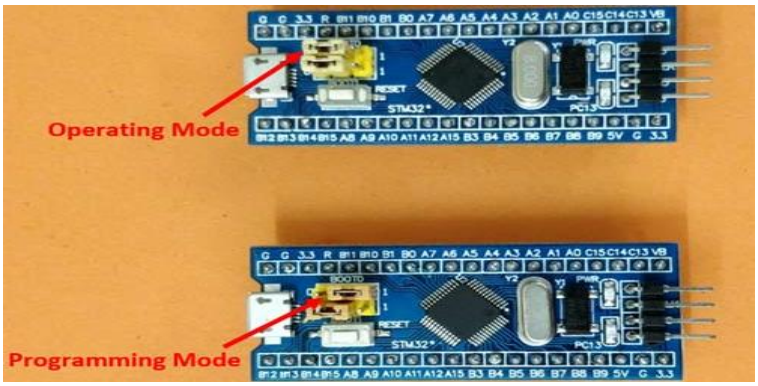


Figura 44. Modos de operación del STM32.

Tomado de (Raj, 2018).

3.6. Configuración interna del STM32F103

Expuestos los programas anteriores, se abre STM32CubeMx y se crea un nuevo proyecto, seleccionando la placa STM32 que se utilizará, la cual es: STM32F103C8Tx. Después, la aplicación abrirá su interfaz la cual tiene tres pestañas relevantes donde se trabajará y que se detallan en seguida.

3.6.1. Pinout & Configuration

Permite configurar los pines del STM32, habilitar funciones y modificar varios registros de forma gráfica.

Primeramente, se procede a establecer la comunicación del microcontrolador la cual utilizará USART 2 para el enlace con el Arduino Nano y USART 3 para la transmisión de información en las pruebas. Entonces, se debe abrir las siguientes funciones y modificar lo que se menciona a continuación:

- **Reset and Clock Controller (RCC**): Gestiona el reloj del microcontrolador y su sistema de reinicio (STMicroelectronics, 2018). Se establece el siguiente valor en la configuración:
    - **High Speed External Clock (HSE):** Crystal / Ceramic Resonator.
- **USART 2 y USART 3:** Son puertos seriales que pueden ser programados para ser de tipo síncronos o asíncronos (STMicroelectronics, 2018). A continuación, se establecen los siguientes valores en las configuraciones de ambas interfaces:
    - **Mode:** Asyncronous.
    - **DMA Request (en USART2):** USART2_RX.
    - **NVIC Interrupt Table – “USART2 global interrupt”:** Enabled.
    - **Baud Rate:** 115200 Bits/s.
    - **DMA Request (en USART3):** USART3_TX.
    - **NVIC Interrupt Table – “USART3 global interrupt”:** Enabled.

Para el movimiento de los servomotores que corresponden a los dedos de la mano robótica, se procede a utilizar cinco señales PWM (Pulse Width Modulation) de las cuales cuatro provienen del TIMER 3 (canales 1, 2, 3 y 4), y una proviene del TIMER 2 (canal 2). A continuación, se configura las siguientes funciones:

- **TIM2 y TIM3:** Son temporizadores que poseen cuatro canales independientes cada uno y que pueden ser utilizados para diversos propósitos, uno de ellos es la generación de señales PWM las cuales son esenciales para el control de los servomotores (STMicroelectronics, 2018). Dicho esto, se aplican las siguientes configuraciones en cada temporizador:
    - **Clock Source:** Internal Clock.
    - **Channel2 (en TIM2):** PWM Generation CH2.
    - **Channel1 (en TIM3):** PWM Generation CH1.
    - **Channel2 (en TIM3):** PWM Generation CH2.
    - **Channel3 (en TIM3):** PWM Generation CH3.
    - **Channel4 (en TIM3):** PWM Generation CH4.
    - **Prescaler (PSC – 16 bits value):** 720 – 1.
    - **Counter Period (Auto Reload Register – 16 bits value):** 1000 – 1.

Los valores ingresados del Prescaler y del número de conteos (Counter Period) fueron obtenidos teniendo en cuenta que un servomotor genérico trabaja a una frecuencia de 50 Hz equivalente a 20 milisegundos de periodo y que los temporizadores TIM2 y TIM3 operan bajo el Bus Periférico Avanzado (APB) a una frecuencia de 36 MHz.

Lo que implica que se debe ingresar valores que cumplan con estas condiciones de periodo PWM para un óptimo funcionamiento de los servomotores (STMicroelectronics, 2018).

Para ello se utiliza la siguiente ecuación:

$$T = (\#Cont + 1) * \left(\frac{1}{f}\right) * (\mathrm{Pr}) \qquad \text{(Ecuación 10)}$$

Donde:

T: Periodo de la señal PWM.

#Cont+1: Número de conteos.

f: Frecuencia de 36 MHz.

Pr: Prescaler.

Resolviendo la ecuación con los valores planteados se obtiene el siguiente resultado que cumple con las condiciones necesarias:

$$T = (999 + 1) * \left(\frac{1}{36x10^6}\right) * (719)$$

$$T = (1000) * \left(\frac{1}{36x10^6}\right) * (719)$$

$$T = 19{,}97\ ms$$

Finalmente, se procede a configurar los pines de los LEDs indicadores. Para ello, se debe posicionarse con el cursor sobre cualquier pin del procesador y seleccionar la opción: GPIO_Output, para establecerlo como salida o entrada general (STMicroelectronics, 2018).

En este caso como se pudo observar en la Figura 43, se utilizarán los pines: PA15, PB3, PB4 y PB5 (el pin PC13 corresponde al LED que se encuentra en la placa STM32).

A continuación, en la Figura 45 se muestra el procesador configurado completamente.

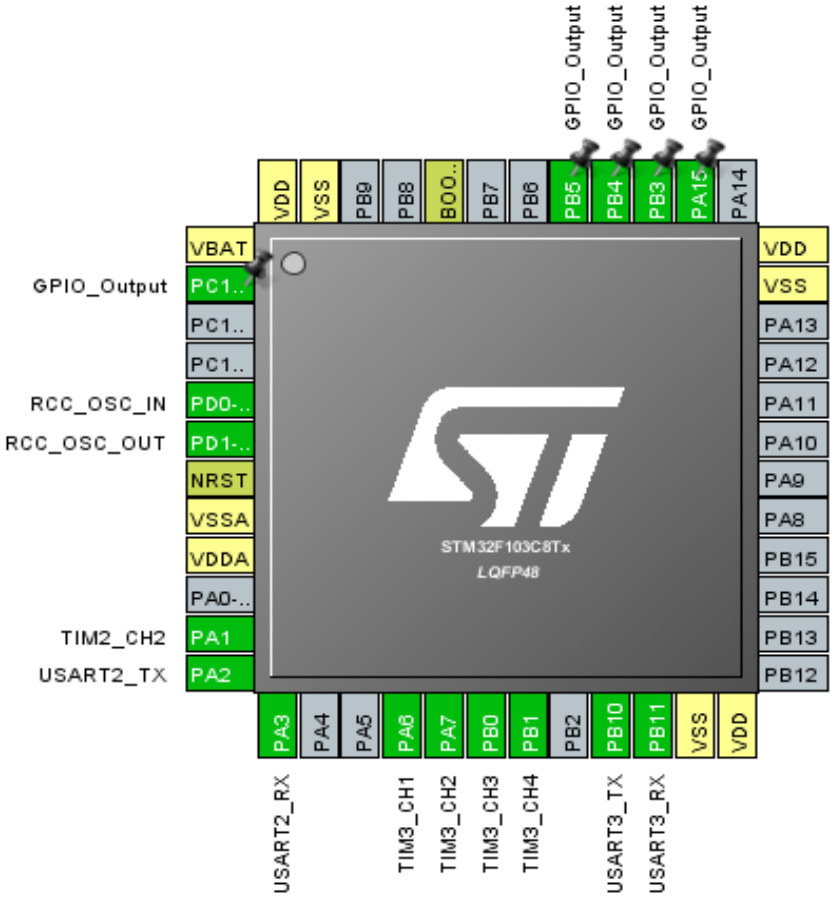


Figura 45. STM32F103C8Tx configurado.

3.6.2. Clock Configuration y Project Manager

Por un lado, Clock Configuration permite configurar la velocidad del reloj del procesador, es decir, se puede establecer qué tan rápido será y su capacidad de ahorro de energía y recursos.

Por otra parte, en Project Manager se tiene las opciones modificables para generar el código C del proyecto que se está realizando.

Dicho esto, en seguida se exponen los siguientes puntos que deben ser configurados en Clock Configuration para tener un buen rendimiento del procesador:

- **PLL (Phase – Locked Loop):** Es un circuito lógico el cual adquiere una frecuencia de reloj de entrada y la multiplica para tener una frecuencia de salida más alta.

  Tiene dos tipos de señales entrantes: HSI y HSE (Embedded, 2016).

  - **Multiplexor de Origen PLL:** Se debe escoger la opción HSE, que posee una frecuencia de 8MHz.
  - **PLLMul:** Es un valor que corresponde al número por el cual se multiplicará la señal entrante, se procede a escoger X9 para obtener una señal PLL de 72 MHz.
- **Multiplexor del Reloj del Sistema:** Se escoge la opción PLLCLK para indicar que se utilizará el reloj PLL, deshabilitando aquellos sin uso y por ende, ahorrando energía (Embedded, 2016).
- Como se mencionó anteriormente, TIM2 y TIM3 utilizan el Bus Periférico Avanzado (APB) el cual debe tener una frecuencia de 36MHz para generar un periodo de señal PWM de 20 ms.

  Para que esto suceda, en el Prescaler APB1 se establece un valor de 4 para poder dividirlo con la señal entrante (HCLK) de 72 MHz, obteniendo una de 18 MHz que posteriormente es multiplicada por 2, resultando la frecuencia deseada para el reloj de los temporizadores que generarán un periodo de señal PWM que trabaje con los servomotores genéricos.

En la Figura 46, se puede apreciar el esquema con lo anteriormente explicado para una mejor comprensión.

Finalmente, en la pestaña Project Manager se inserta el nombre del proyecto, la ubicación donde se guardará y en la opción “Toolchain / IDE” se selecciona: MDK – ARM V5, para que este genere el proyecto que será abierto automáticamente en Keil uVision 5. Es importante mencionar que cada vez que se actualice el proyecto de STM32CubeMX, se aplicarán cambios de forma automática en Keil uVision 5 ya que estos dos programas están enlazados entre sí.

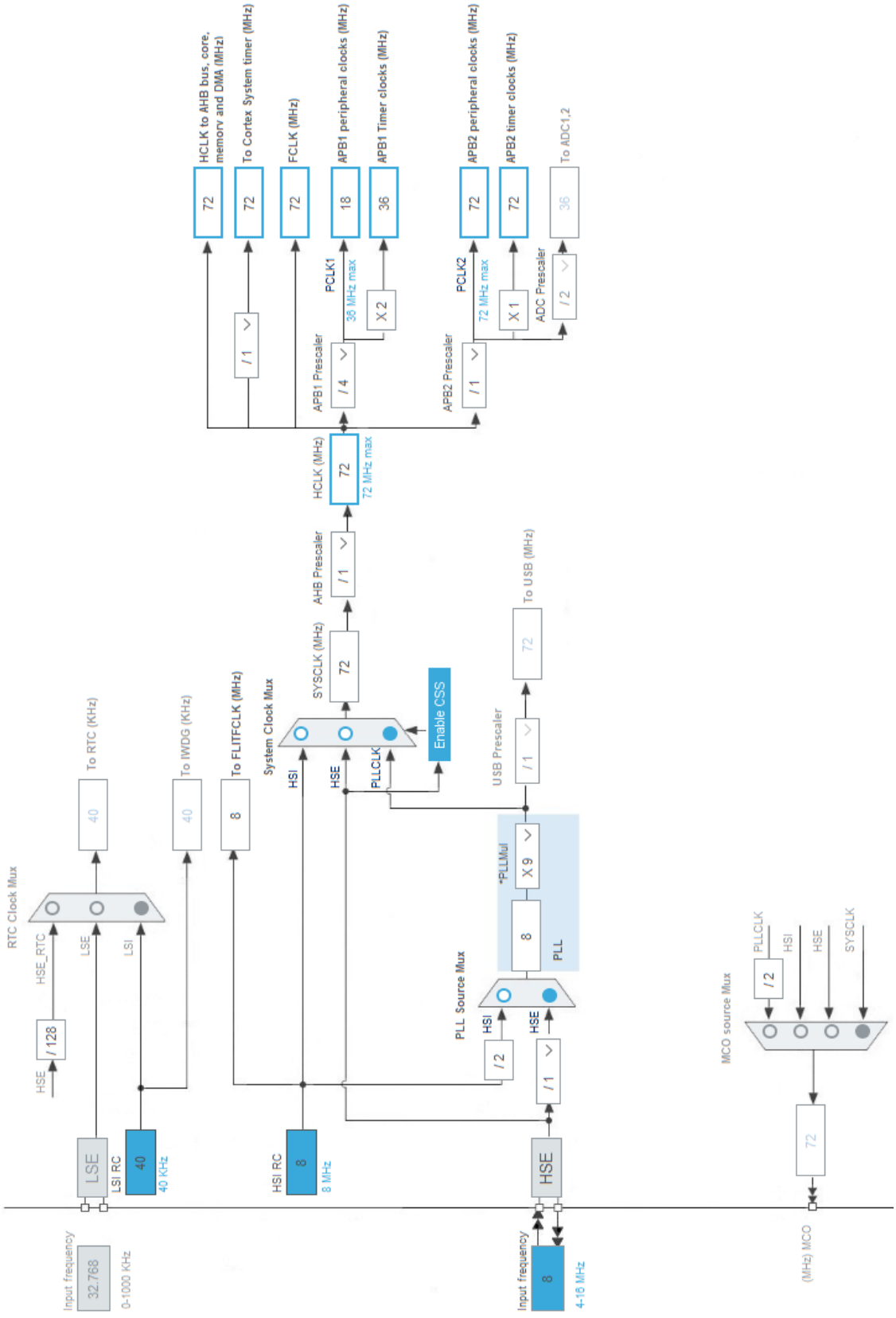


Figura 46. Clock Configuration.

3.7. Procesamiento de las señales y aplicación de Naïve Bayes

Una vez se haya conectado los componentes y configurado el microcontrolador, se procede a trabajar sobre el proyecto generado en Keil uVision 5 en el archivo “main.c” que posee código escrito de forma predeterminada que no puede ser manipulado o borrado ya que corresponde a las modificaciones realizadas anteriormente, por lo cual se debe programar únicamente en las áreas señaladas.

Por otro lado, es importante mencionar que previo al programa principal, se tiene que incluir la librería “math.h” la cual sirve para realizar operaciones matemáticas. Además, se debe definir los valores aproximados de PI y Euler: M_PI = 3.14159 y M_E = 2.71828. Finalmente, se debe declarar las siguientes variables expuestas en la Tabla 3.

Tabla 3.

Variables globales.

Variables de comunicación serial	
Tipo	**Nombre**
uint8_t	Rx_data[1]
int	Rx_dataT
uint8_t	rxData_vect[800]
uint8_t	Rx_Trans[3]
uint8_t	Rx_TransNegative[4]
Variables auxiliares	
int	filas, columnas
uint8_t	msg_aux[1]
uint8_t	msg_fin[22]
uint8_t	recibir
int	contador

uint8_t	listo
uint8_t	miles, centenas, decenas, unidades
uint8_t	milesN, centenasN, decenasN, unidadesN
	Variables de extracción de características (features)
float	mediaP[8]
float	STmedia[8]
float	varianzaT[8]
char	msgMedia[30], msgVar[30], msgDesvEst[30], msgRMS[30]
char	msgFeatures[5]
float	features[5]
	Variables de procesamiento
int	mayorSensores[8]
int	numParcial
float	NumParcialST
char	msgStandard[6]
	Variables de Naïve Bayes
float	dataTraining[128]
float	openT[2][32], closeT[2][32]
float	conditionalP, feature, CPAcumOpen, CPAcumClose, priorProbability = 0.5, evidenceOpen, evidenceClose, pAbierto, pCerrado, evidencia
char	msgAbierto[10], msgCerrado[10]
	Variables de control
int	limitador = 30, datos = 240
int	ventana[100][8], absVentana[100][8]
float	aStandard[100][8], aStandardABS[100][8]

3.7.1. Función: main

Es la función principal del programa que puede ser apreciada en el diagrama de flujo expuesto en la Figura 47. En esta se realiza la inicialización de los temporizadores que originarán las PWM, así como también la interrupción de recepción.

Posee un ciclo while el cual se repetirá constantemente mientras el microcontrolador se encuentre encendido. A continuación, se explica brevemente sobre ciertas funciones relevantes del diagrama.

"HAL_UART_Receive_IT(&huart2, Rx_data, sizeof(Rx_data))"

"HAL_UART_Transmit(&huart3, msg_fin, sizeof(msg_fin), 100)"

"HAL_UART_Transmit(&huart2, msg_aux, sizeof(msg_aux), 100)"

Donde:

- **HAL_UART_Receive_IT:** Función para iniciar / reiniciar la interrupción de recepción.
- **HAL_UART_Transmit:** Función para enviar una cadena uint8_t por el serial.
- **&huart2:** Es un puntero que indica que la interrupción se aplicará en el puerto serial 2.
- **&huart3:** Es un puntero que indica que la transmisión se realizará en el puerto serial 3.
- **Rx_data:** Es un vector uint8_t de un elemento que guardará el valor recibido.
- **msg_fin:** Es un vector el cual contiene una cadena de texto que dice "Comienza nuevo vector" para indicar que se reinicia el ciclo.
- **msg_aux:** Es un vector auxiliar de un elemento en donde se asigna un valor que se puede transmitir por el serial 2 o por el serial 3.
- **Sizeof():** Corresponde al tamaño de la variable que se planea enviar o recibir.

- **100:** Corresponde a la duración del envío expresado en milisegundos.

"HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2)"

"HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1 / 2 / 3 / 4)"

Donde:

- **HAL_TIM_PWM_Start:** Función para iniciar la generación PWM.
- **&htim2 / &htim3:** Son punteros que indican que la PWM se originará por los temporizadores 2 y 3.
- **TIM_CHANNEL_2:** Corresponde al canal 1 del temporizador 2.
- **TIM_CHANNEL_1 / 2 / 3 / 4:** Corresponden a los canales 1, 2, 3 y 4 del temporizador 3.

"(pAbierto > pCerrado) && (pAbierto >= 0.60)"

"(pCerrado > pAbierto) && (pCerrado >= 0.60)"

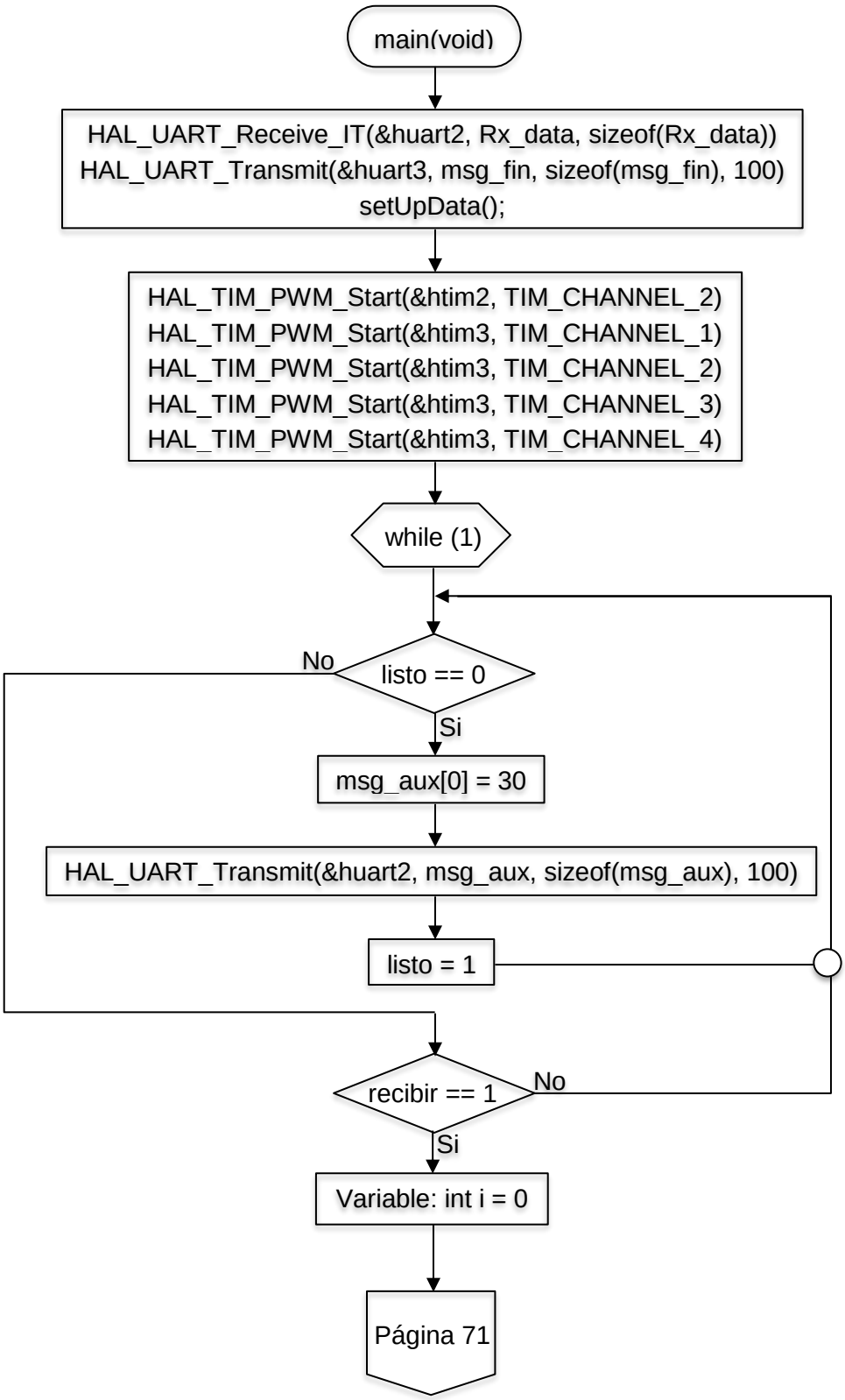
Donde:

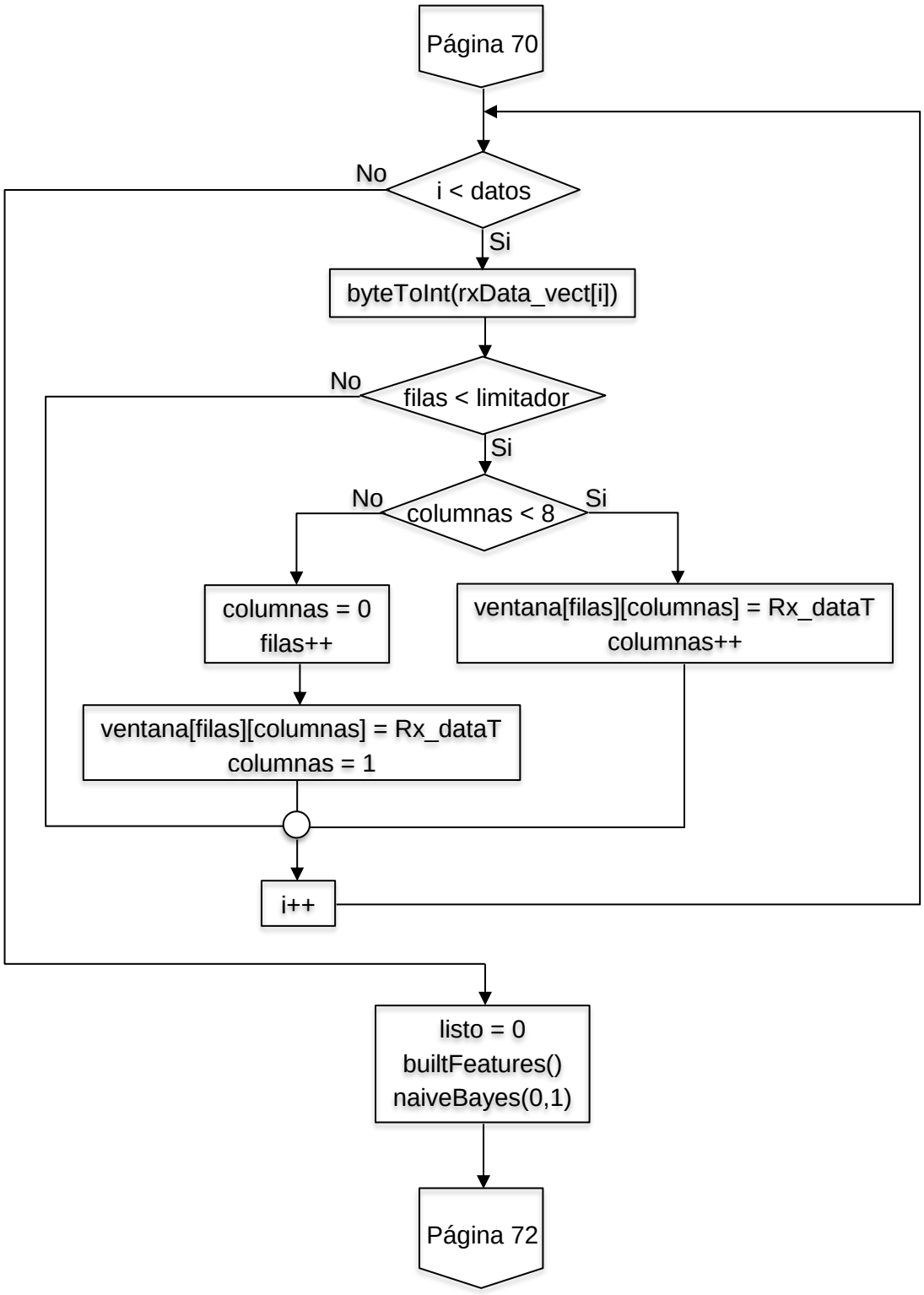
- **pAbierto y pCerrado:** Variables que se obtienen en la función naiveBayes y que poseen la probabilidad de abierto y de cerrado del clasificador.
- **0.60:** Corresponde al umbral de 60% que será comparado para predecir el movimiento.

"HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_4 / 5 / 3 / 15)"

Donde:

- **HAL_GPIO_TogglePin:** Función para cambiar el estado de un pin y así encender un LED.
- **GPIOB:** Indicador del grupo al que pertenece el pin.
- **GPIO_PIN_4 / 5 / 3 / 15:** Número de los pines que se va a utilizar.


main(void)
HAL_UART_Receive_IT(&huart2, Rx_data, sizeof(Rx_data))
HAL_UART_Transmit(&huart3, msg_fin, sizeof(msg_fin), 100)
setUpData();
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2)
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1)
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_2)
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_3)
HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_4)
while (1)
No
listo == 0
Si
msg_aux[0] = 30
HAL_UART_Transmit(&huart2, msg_aux, sizeof(msg_aux), 100)
listo = 1
recibir == 1
No
Si
Variable: int i = 0
Página 71


Página 70
No
i < datos
Si
byteToInt(rxData_vect[i])
No
filas < limitador
Si
No
columnas < 8
Si
columnas = 0
filas++
ventana[filas][columnas] = Rx_dataT
columnas++
ventana[filas][columnas] = Rx_dataT
columnas = 1
i++
listo = 0
builtFeatures()
naiveBayes(0,1)
Página 72

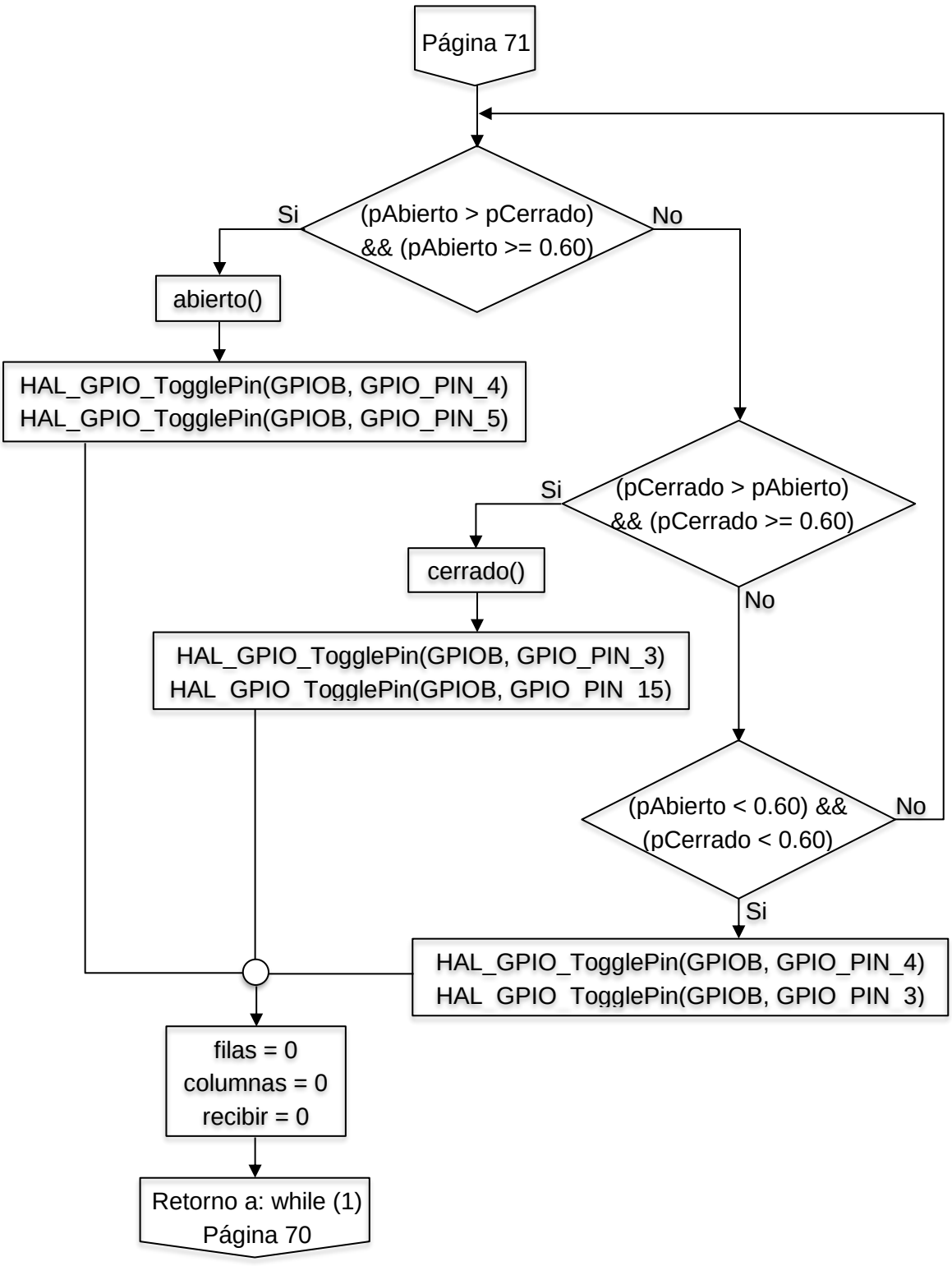


Figura 47. Función principal main.

3.7.2. Función: setUpData

Inicialmente, se mencionó que el algoritmo clasificador Naïve Bayes opera con un entrenamiento previo el cual es realizado en el Software Matlab.

Producto de esto, se tiene una serie de características que corresponden a las medias y desviaciones estándar tomadas ya sea, en una persona o en un grupo de personas.

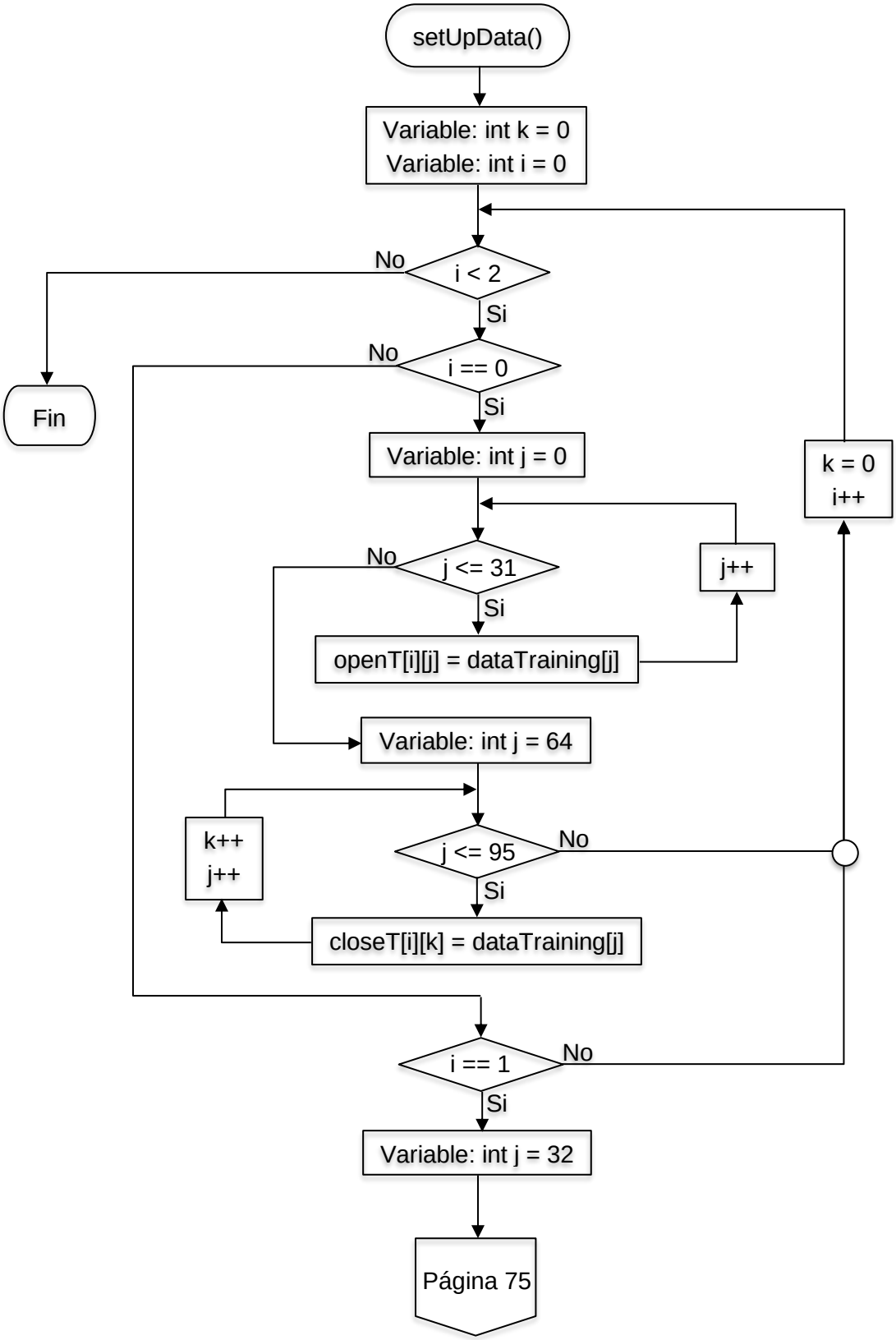
Estos valores se los debe insertar manualmente en el código mediante el vector dataTraining que posee un tamaño para 128 datos flotantes los cuales se dividen de la siguiente forma:

- **Desde el 0 al 31:** Valores medios del movimiento abierto.
- **Desde el 32 al 63:** Valores de la desviación estándar del movimiento abierto.
- **Desde el 64 al 95:** Valores medios del movimiento cerrado.
- **Desde el 96 al 127:** Valores de la desviación estándar del movimiento cerrado.

Dicho esto, la función setUpData funciona para distribuir las características de abierto y cerrado del vector dataTraining en dos matrices bidimensionales: openT y closeT que poseen un tamaño de 2 filas por 32 columnas.

La primera fila es para los valores medios y la segunda es para las desviaciones estándar.

A continuación en la Figura 48, se presenta el diagrama de flujo de esta función detallando como se realiza el proceso.


setUpData()
Variable: int k = 0
Variable: int i = 0
No
i < 2
Si
No
i == 0
Si
Fin
Variable: int j = 0
k = 0
i++
No
j <= 31
j++
Si
openT[i][j] = dataTraining[j]
Variable: int j = 64
k++
j++
j <= 95
No
Si
closeT[i][k] = dataTraining[j]
i == 1
No
Si
Variable: int j = 32
Página 75

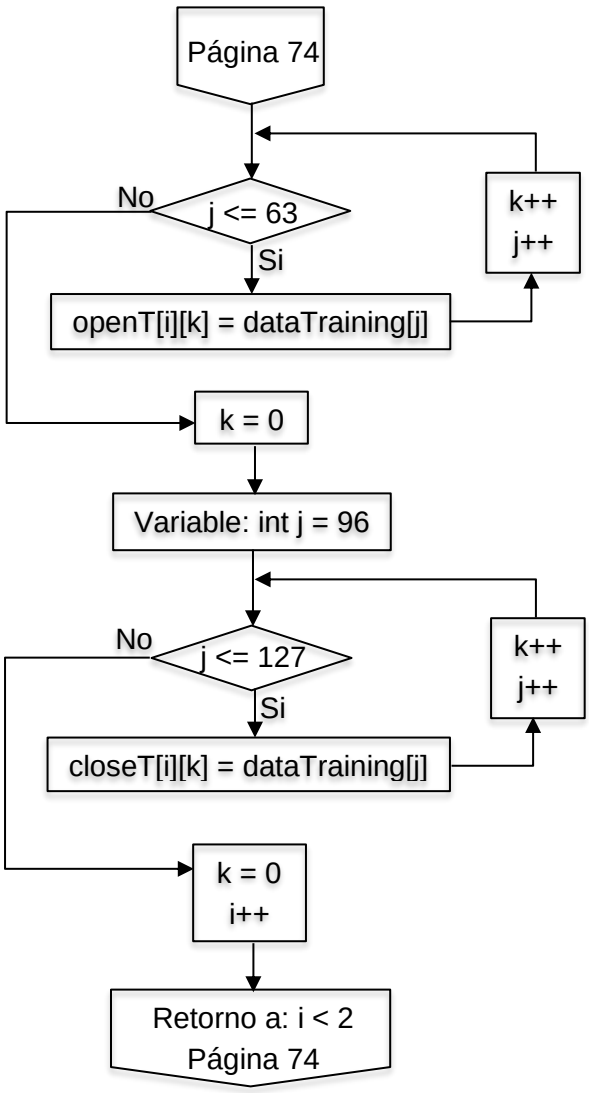


Figura 48. Función setUpData.

3.7.3. Función: HAL_UART_RxCpltCallback

Es una función de Keil la cual se ejecuta cuando se produce una interrupción de recepción y se la puede observar de mejor manera en la Figura 49.

Posee un parámetro de entrada “UART_HandleTypeDef *huart” el cual por defecto, corresponde al puntero de la interfaz serial en donde está posicionada la interrupción.

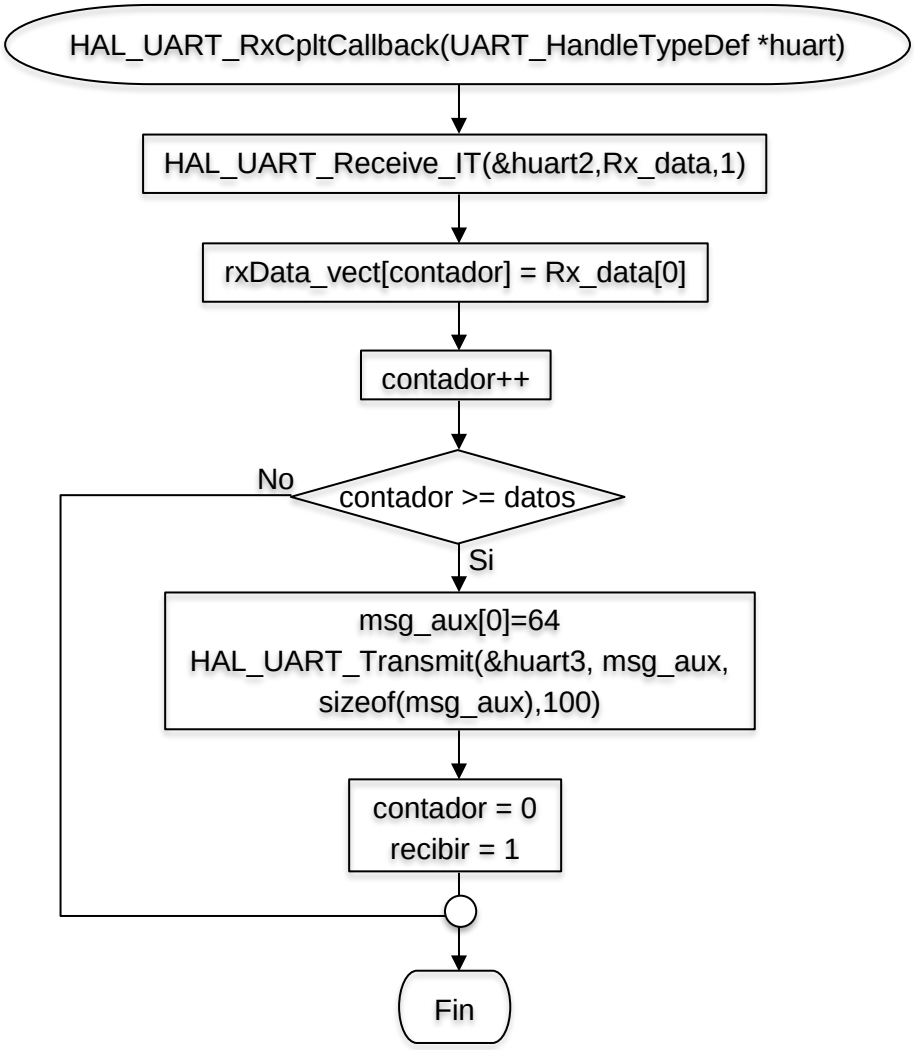


Figura 49. Función HAL_UART_RxCpltCallback.

3.7.4. Función: byteToInt

El sensor MYO Armband captura y transmite números enteros entre -128 y 128 al Arduino Nano que son enviados en formato byte (de 0 a 255) al STM32 por la función printData. Los valores negativos llegan en complemento a dos por lo que es necesario aplicar una máscara de 256 para recuperar el número original.

La función byteToInt es utilizada para implementar este proceso. Tiene como parámetro de entrada un byte (uint8_t) y si este es mayor a 128, aplica la máscara

restándole 256, resolviendo el número de origen. La estructura de esta función se puede observar en la Figura 50.

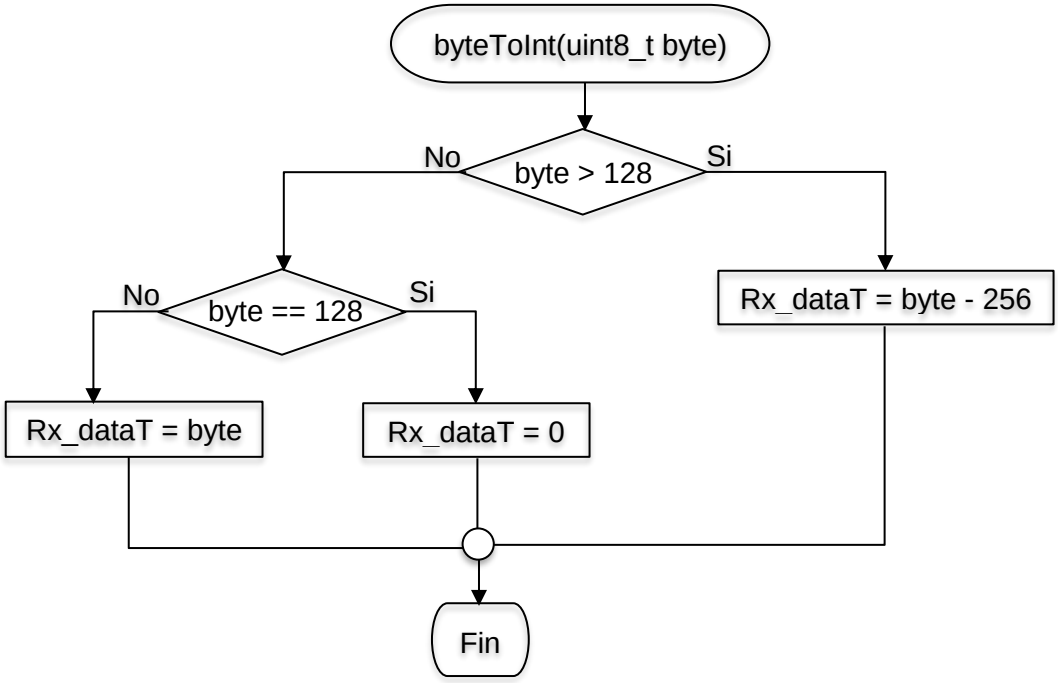


Figura 50. Función byteToInt.

3.7.5. Función: builtFeatures

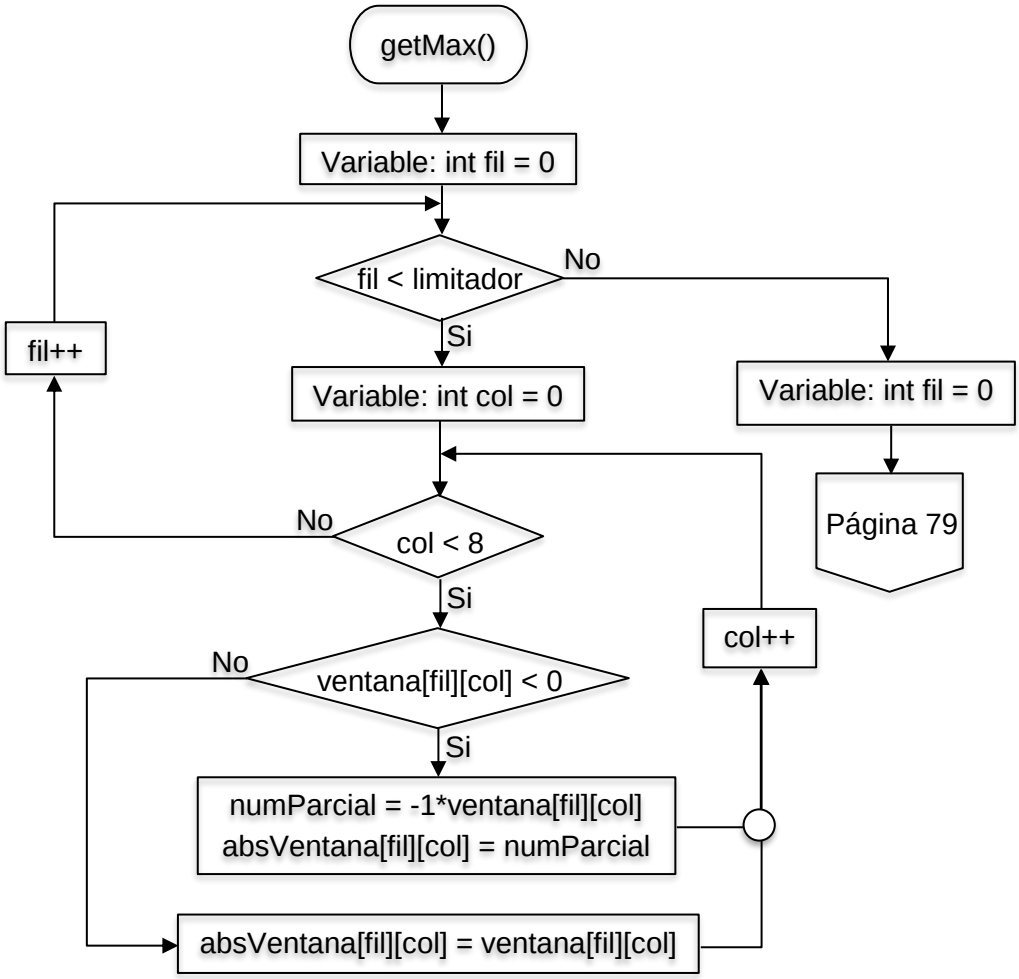
En builtFeatures se tiene la llamada de las funciones para obtener las características de los valores del vector ventana que posteriormente serán utilizados en el clasificador. Se la implementó únicamente para reducir el código en la función main. A continuación, se enlista las funciones que son llamadas:

- getMax()
- standarization()
- getMediaABS()
- getVarianza()
- getDesviacionEstandar()
- getRMS()

3.7.5.1. Función: getMax

Esta función se utiliza para obtener los valores más altos de cada señal electromiográfica. Para ello se procede a transformar todos los números negativos del vector ventana en valores absolutos (multiplicándolos por -1), los cuales posteriormente son almacenados en un nuevo arreglo bidimensional llamado: absVentana.

Después, los valores de este nuevo vector son comparados entre sí y el mayor de cada señal es almacenado en la matriz unidimensional mayorSensores. Para entender lo anteriormente explicado de mejor manera, en la Figura 51 se presenta el diagrama de flujo de la función.



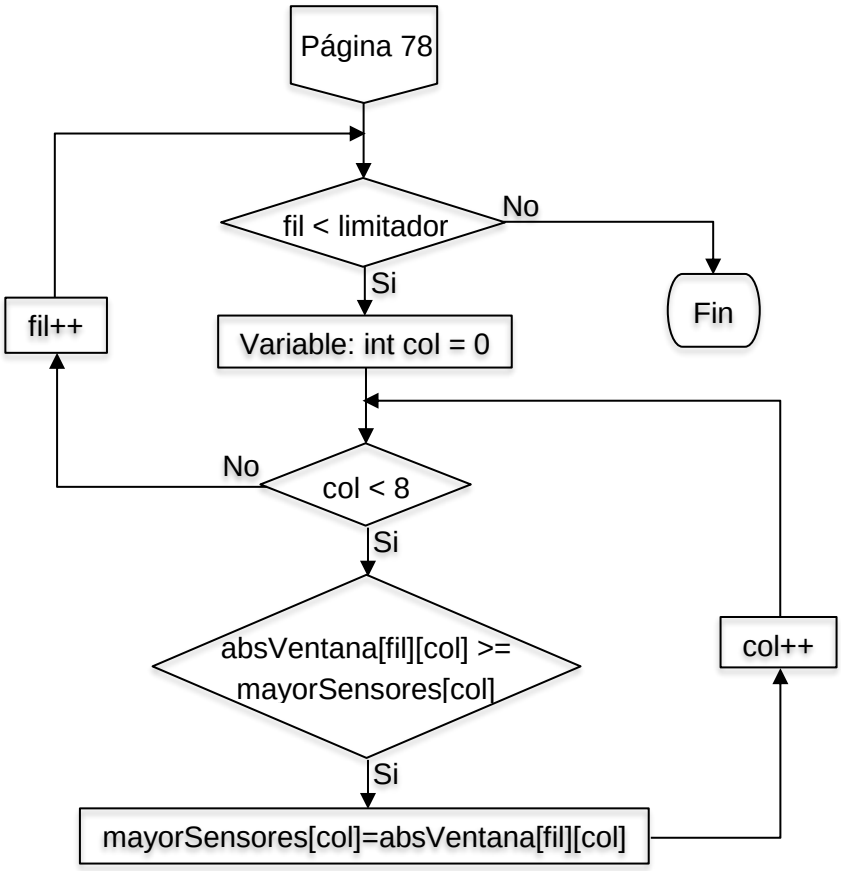


Figura 51. Función getMax.

3.7.5.2. Función: standarization

Es la función encargada de normalizar los datos del vector ventana para que estos se encuentren en números flotantes entre 0 y 1. Para que esto sea posible se debe dividir el valor de cada señal para su número máximo y posteriormente, guardar el resultado en la nueva matriz bidimensional: aStandard, como se muestra en la Figura 52.

Se debe mencionar que se realiza la normalización para poder obtener las características a partir de estos valores.

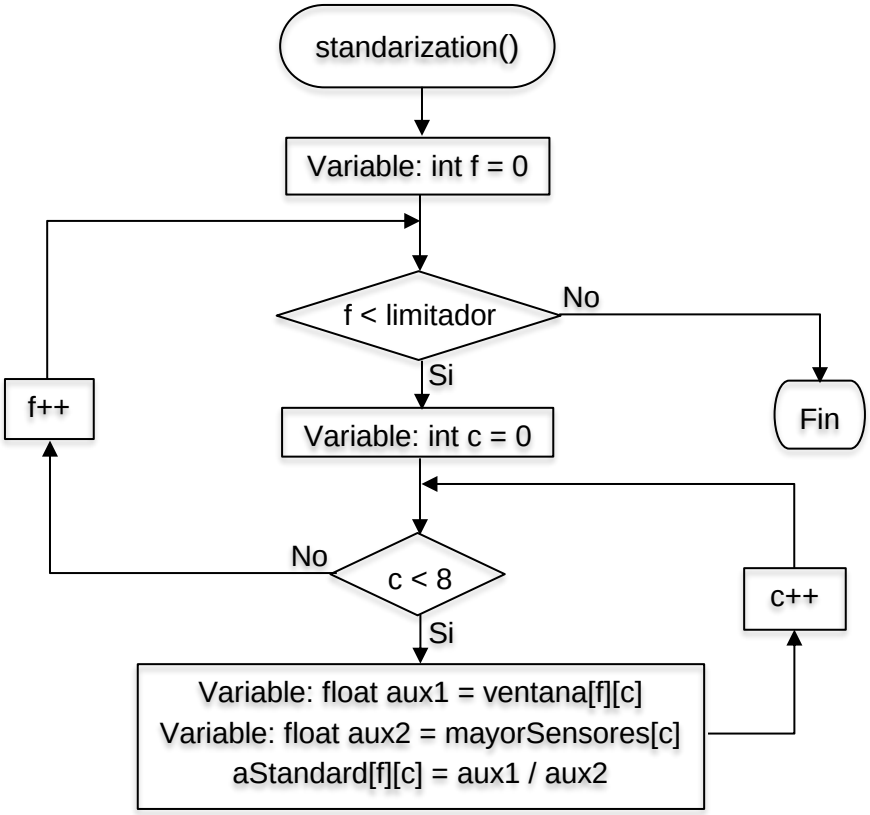


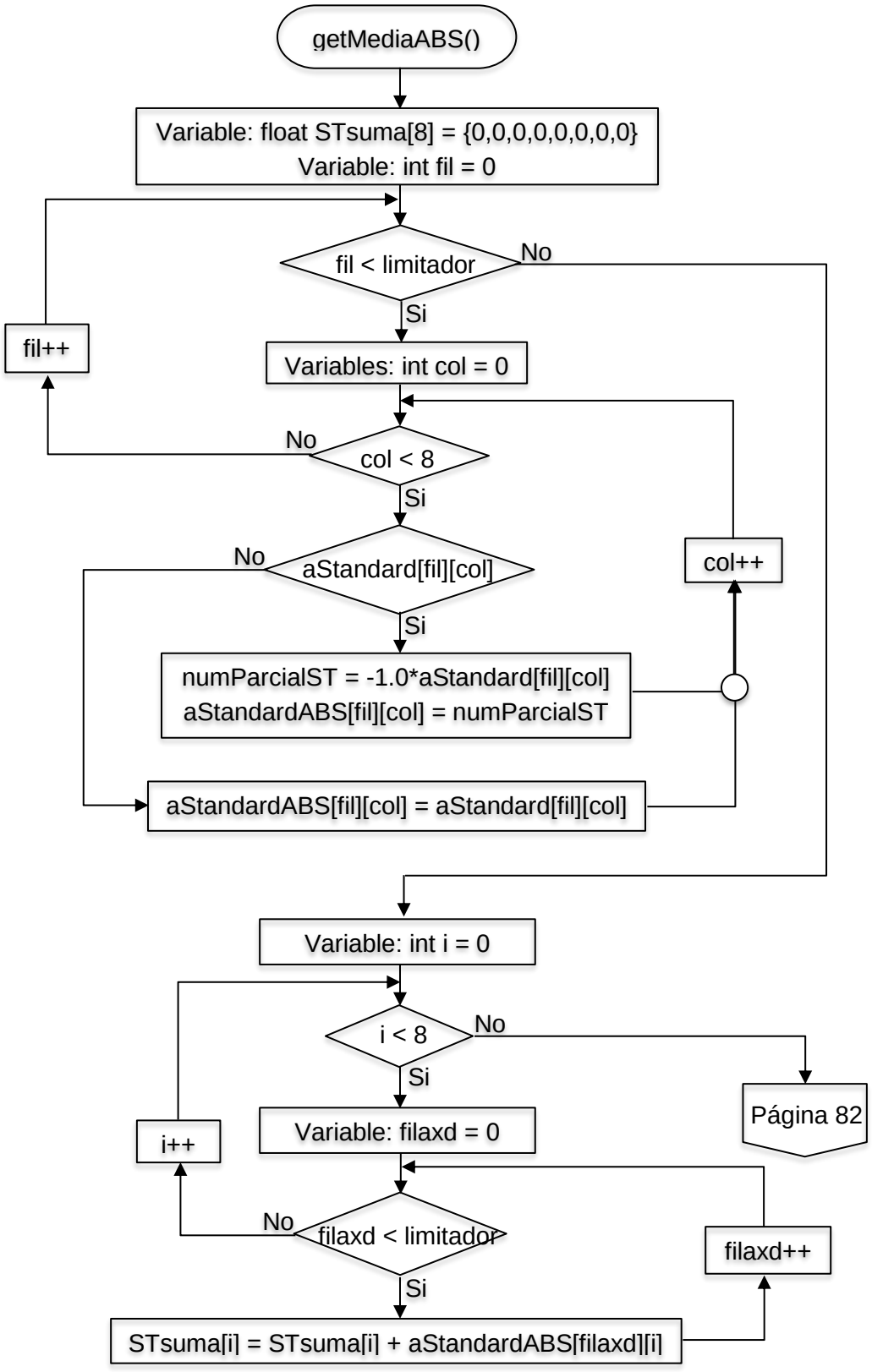
Figura 52. Función standarization.

3.7.5.3. Función: getMediaABS

Esta función corresponde al cálculo del MAV o Valor Medio Absoluto y es la primera de las características que se procede a obtener y se puede observar un diagrama de flujo de esta en la Figura 53.

Para lograr obtener estas características, primeramente se transforma los valores del vector aStandard en números absolutos y se los guarda en la nueva matriz bidimensional: aStandardABS, del mismo modo que se explicó anteriormente.

Después, se realiza una sumatoria de cada valor de cada columna, obteniendo siete resultados que serán divididos para el número total de filas. Finalmente estos valores son ingresados en el arreglo unidimensional: features.


getMediaABS()
Variable: float STsuma[8] = {0,0,0,0,0,0,0,0}
Variable: int fil = 0
fil < limitador
No
Si
fil++
Variables: int col = 0
No
col < 8
Si
No
aStandard[fil][col]
col++
Si
numParcialST = -1.0*aStandard[fil][col]
aStandardABS[fil][col] = numParcialST
aStandardABS[fil][col] = aStandard[fil][col]
Variable: int i = 0
i < 8
No
Si
Página 82
i++
Variable: filaxd = 0
No
filaxd < limitador
filaxd++
Si
STsuma[i] = STsuma[i] + aStandardABS[filaxd][i]

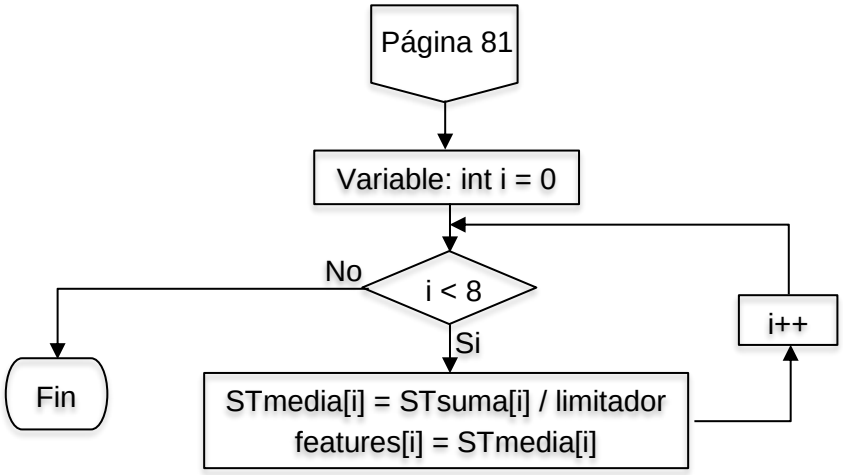
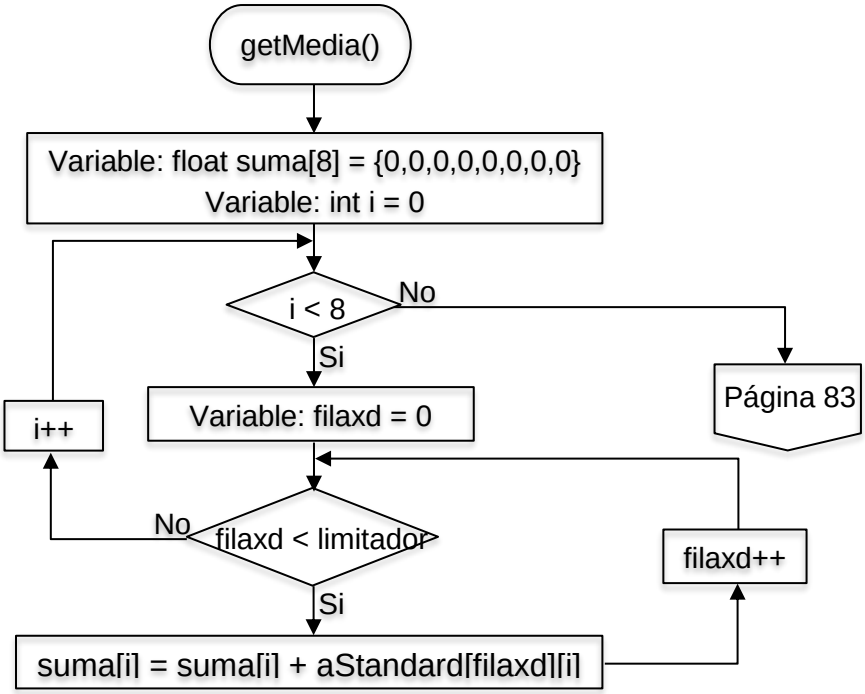


Figura 53. Función getMediaABS.

3.7.5.4. Función: getMedia

Esta función es similar a la anterior, con la única diferencia que ahora los valores medios se obtienen del vector aStandard directamente sin obtener su valor absoluto. El diagrama respectivo de esta función se lo puede encontrar en la Figura 54.



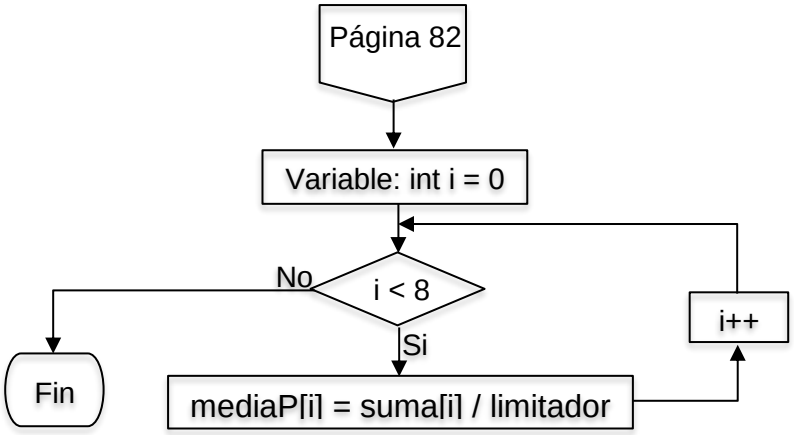
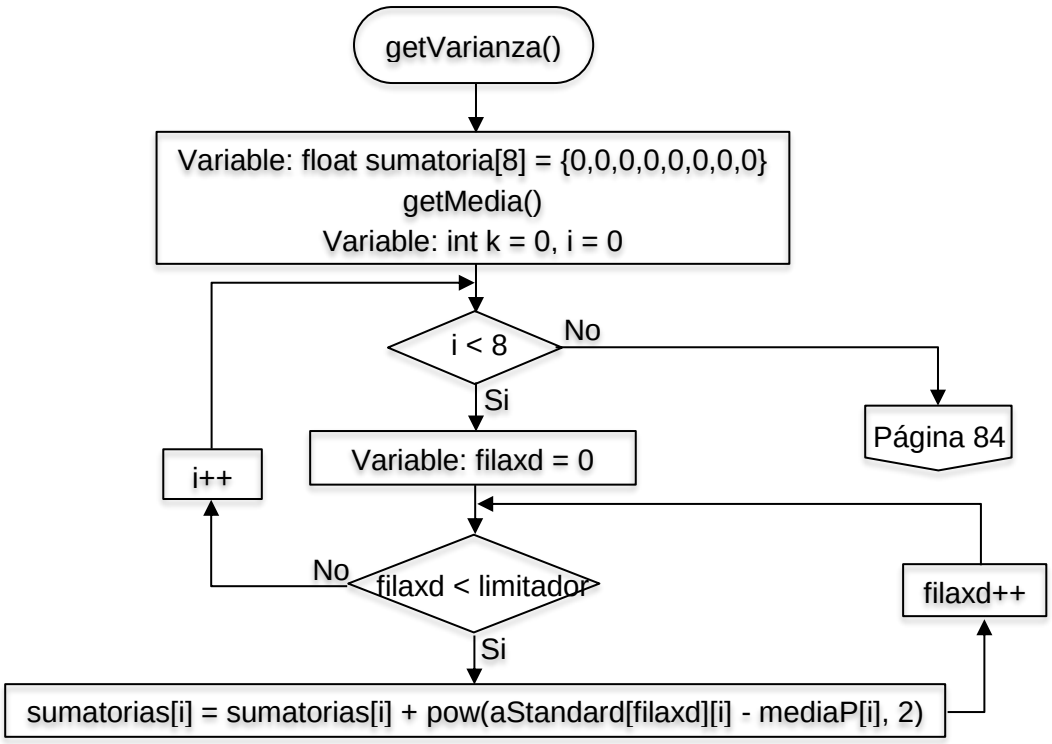


Figura 54. Función getMedia.

3.7.5.5. Función: getVarianza

Esta función corresponde a la aplicación en código C de la ecuación 3 descrita anteriormente en el apartado teórico para la obtención de varianzas de las ocho columnas del vector aStandard que posteriormente, serán guardadas en la matriz unidimensional: features, como se puede apreciar en el diagrama de flujo expuesto en la Figura 55.



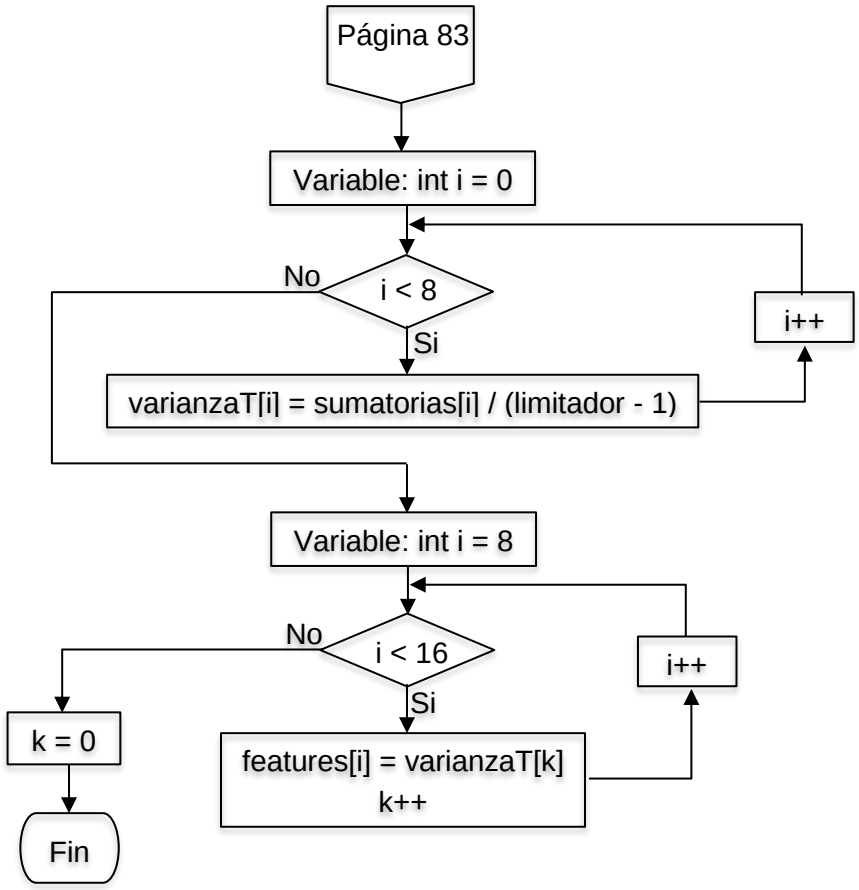


Figura 55. Función getVarianza.

3.7.5.6. Función: getDesviacionEstandar

Al igual que la función anterior, getDesviacionEstandar es una aplicación en código C de la ecuación 4 anteriormente expuesta en el apartado teórico, la cual consiste en obtener la raíz cuadrada (sqrt) de los valores obtenidos en getVarianza.

Dicho esto, a continuación se presenta un diagrama de flujo de la función para una mejor comprensión en la Figura 56 donde se puede apreciar que se utiliza bucles repetitivos “for” para calcular y almacenar los resultados en arreglos de forma ordenada.

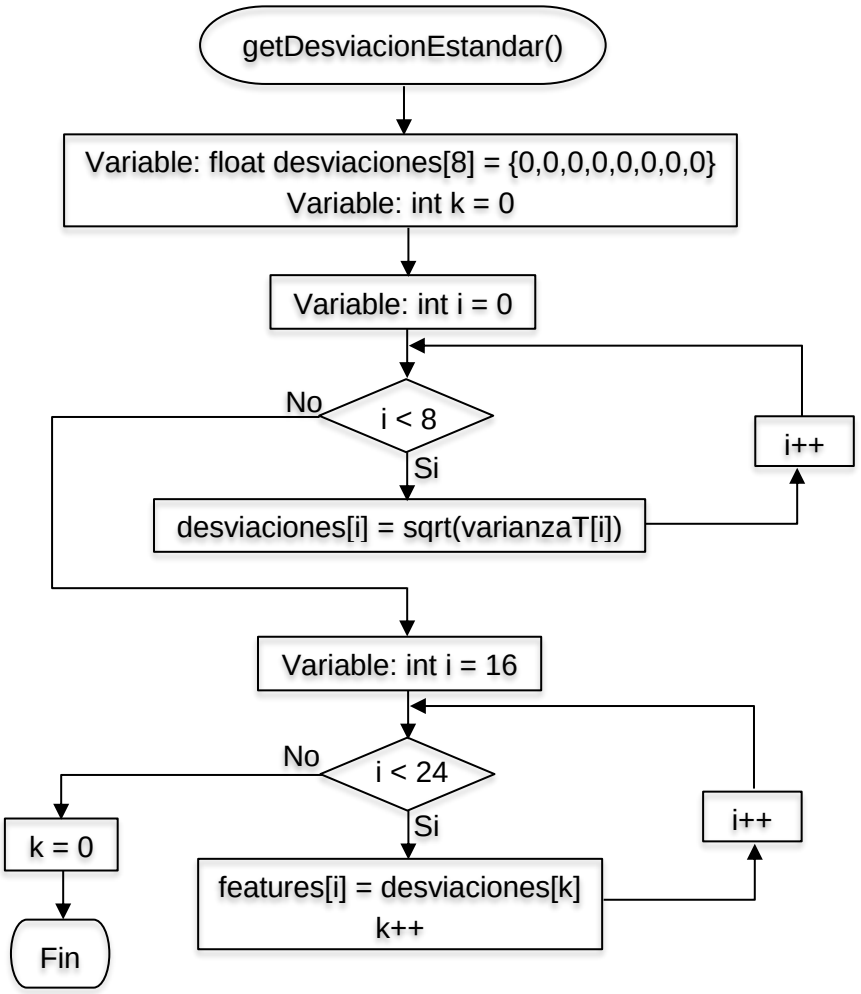


Figura 56. Función getDesviacionEstandar.

3.7.5.7. Función: getRMS

Del mismo modo que se indicó en las funciones anteriores, getRMS es una representación en código C de la ecuación 2 expuesta en el marco teórico y que se puede observar de mejor forma en el diagrama de flujo presentado en la Figura 57.

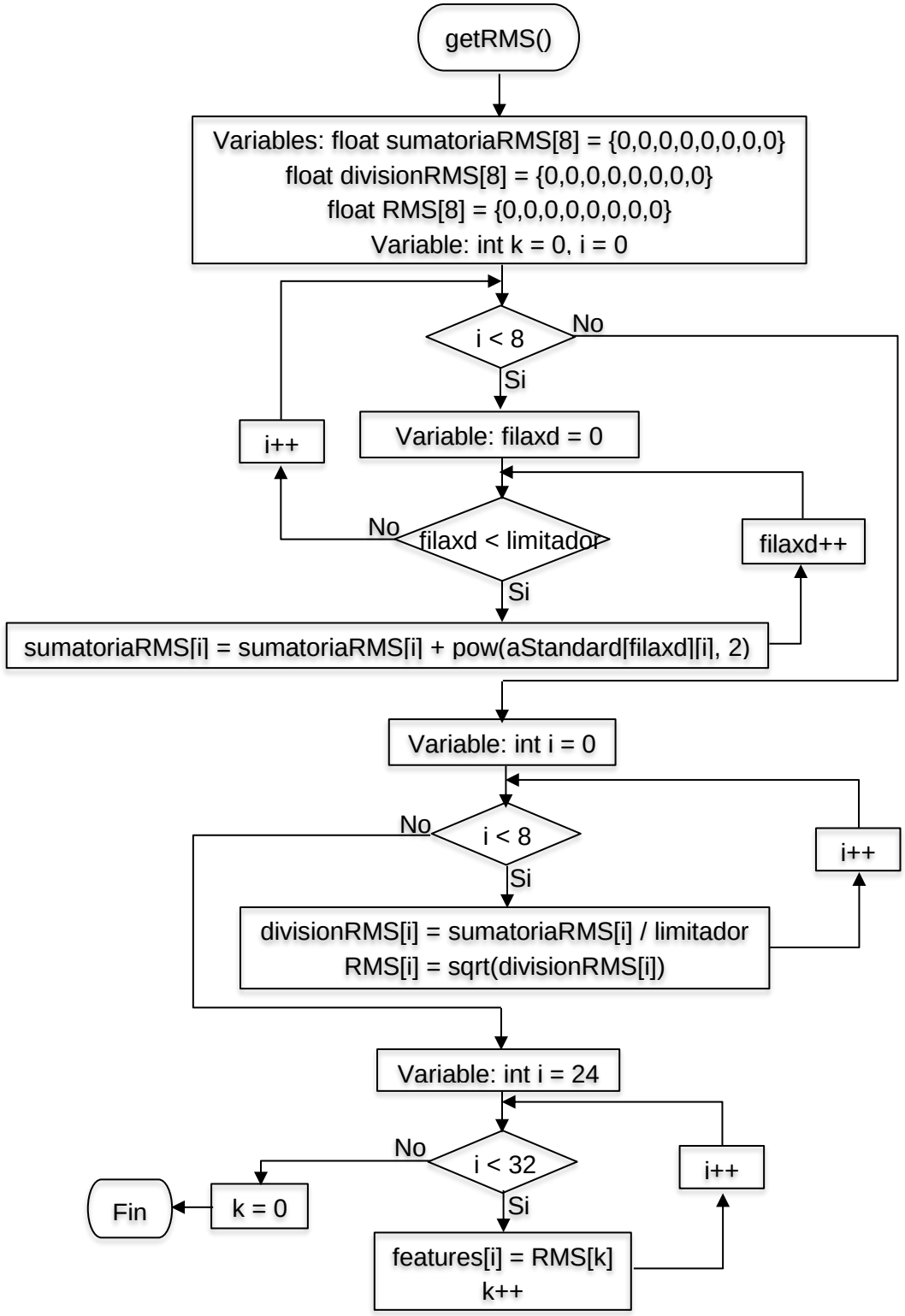


Figura 57. Función getRMS.

3.7.6. Función: conditionalProbability

La presente función tiene parámetros de entrada de tipo float que corresponden a una característica extraída, una desviación estándar y un valor medio del entrenamiento. Esta función es la representación en código de la ecuación 7 la cual para una correcta programación ha sido dividida en partes que pueden observarse en el diagrama de flujo presentado en la Figura 58. También se puede apreciar que en esta función se utiliza los valores definidos al inicio: M_PI y M_E, y que además, se hace uso de los métodos de la librería math.h.

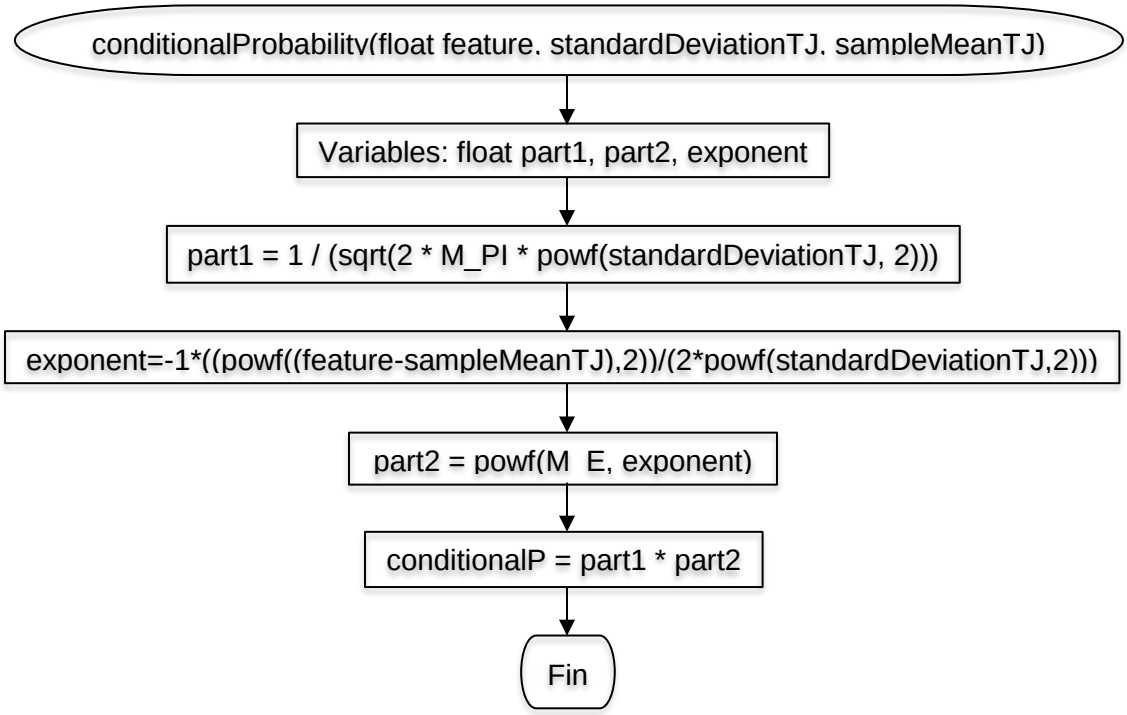


Figura 58. Función conditionalProbability.

3.7.7. Función: cProbabilityAcum

En cProbabilityAcum, se procede a utilizar la función conditionalProbability, consiste en evaluar y acumular probabilidades tanto del movimiento abierto como de cerrado. De igual forma, es una representación de la ecuación 6 detallada y que puede observarse en su diagrama de flujo de la Figura 59.

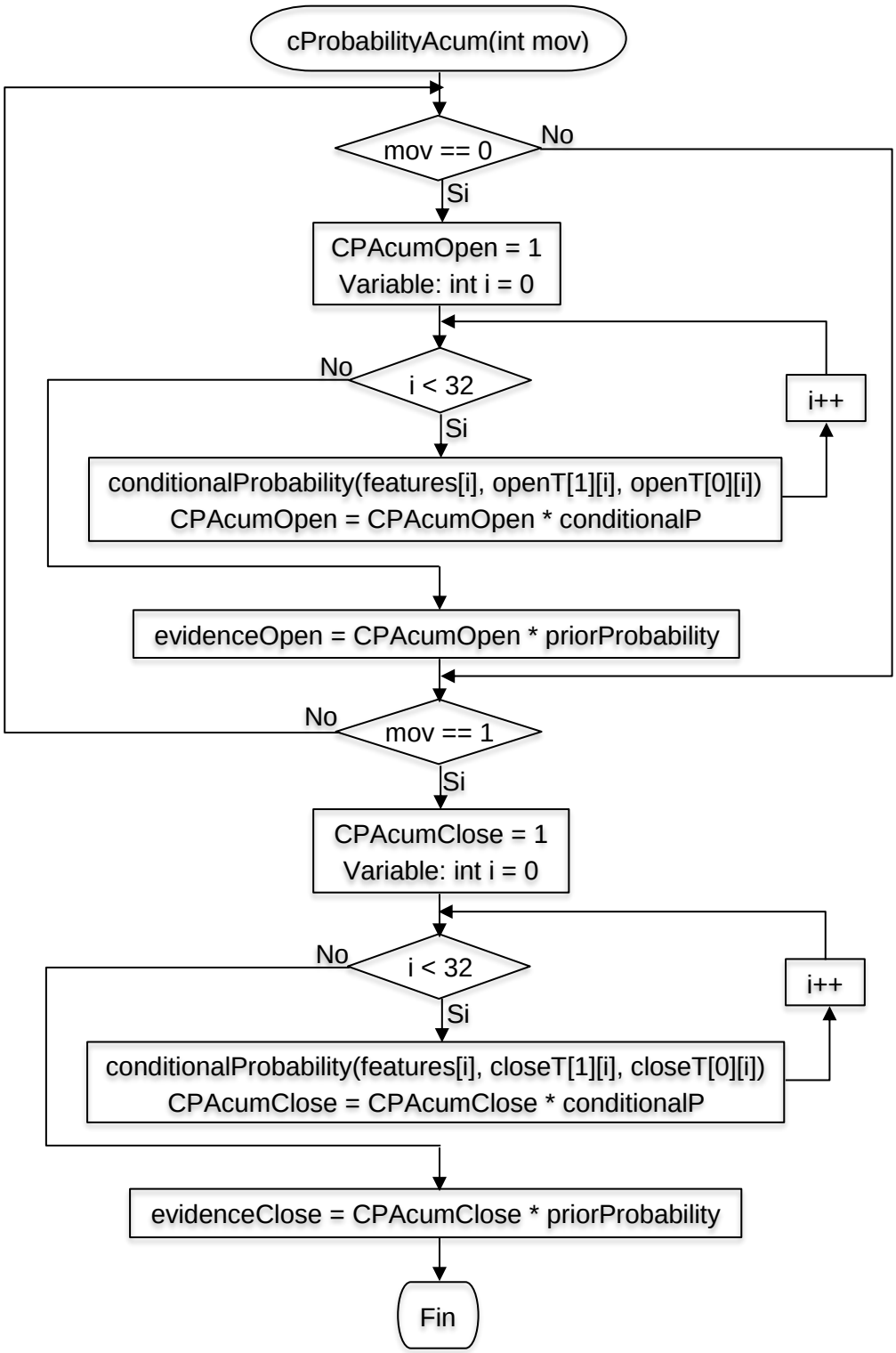


Figura 59. Función cProbabilityAcum.

3.7.8. Función: evidence

Esta función tiene una sola línea de código en la cual se realiza la suma de las variables: evidenceOpen y evidenceClose, obtenidas en la función cProbabilityAcum. El resultado se asigna en la variable global: evidencia.

3.7.9. Función: naiveBayes

En naiveBayes se tiene como parámetros de entrada dos números enteros correspondientes a los movimientos abierto y cerrado. Básicamente, se procede a hacer dos llamados a la función cProbabilityAcum enviando los valores de abierto (0) y cerrado (1) respectivamente, además se llama a la función evidence. Finalmente, los resultados obtenidos son divididos y asignados a las variables: pAbierto y pCerrado.

Cabe resaltar que esta función es la ecuación 5 expuesta en el marco teórico. Además, para comprender su estructura de mejor manera se puede observar la Figura 60 que corresponde al diagrama de flujo de esta función.

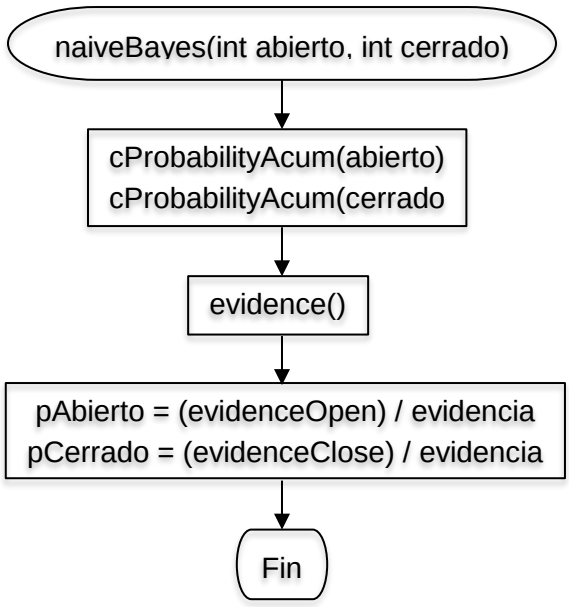


Figura 60. Función naiveBayes.

3.7.10. Funciones: abierto y cerrado

Antes de comenzar, es importante mencionar que un servomotor genérico posee grados de movimiento desde 0° hasta 180° en un periodo de 1 ms a 2 ms (Cheung, 2018), razón por la cual el movimiento de estos está relacionado directamente con el tiempo.

Dicho esto, las funciones que se describen en este apartado son aquellas que modulan las señales PWM para activar los servomotores y realizar la acción de los dedos de la mano robótica.

Para lograr esto, se debe partir de la ecuación 10 la cual es para calcular el periodo de la señal PWM en donde el número de conteos (#Cont) variará para alcanzar el tiempo y los grados adecuados. A continuación, se presenta un ejemplo de lo anteriormente descrito.

$$T = (\#Cont + 1) * \left(\frac{1}{f}\right) * (\mathrm{Pr}) \qquad \text{(Ecuación 10)}$$

$$T = (99 + 1) * \left(\frac{1}{36x10^6}\right) * (719)$$

$$T = (100) * \left(\frac{1}{36x10^6}\right) * (719)$$

$$T = 1{,}99\ ms$$

Se puede observar que con un número de conteos de 100, el periodo de la PWM será de 1,98 ms que aplicando una regla de tres equivale a 178,2°, lo cual indica que el dedo se abrirá hasta su máximo ángulo. El mismo cálculo se aplica a las PWM restantes tanto para abierto como para cerrado. Finalmente, el número de conteos es asignado en el registro CCR (Counter Compare Register).

A continuación, se muestra en las tablas 4 y 5 el valor del CCR por cada PWM.

Tabla 4.

Valores del CCR en los canales PWM para abierto.

Dedo	CCRX	Abierto	Periodo PWM	Ángulo
Pulgar	CCR2	100	1,98 ms	178,2°
Índice	CCR3	90	1,78 ms	160,2°
Medio	CCR3	90	1,78 ms	160,2°
Anular	CCR3	90	1,78 ms	160,2°
Meñique	CCR3	95	1,87 ms	168,3°

Tabla 5.

Valores del CCR en los canales PWM para cerrado.

Dedo	CCRX	Cerrado	Periodo PWM	Ángulo
Pulgar	CCR2	55	1,01 ms	90,9°
Índice	CCR3	60	1,2 ms	108°
Medio	CCR3	60	1,2 ms	108°
Anular	CCR3	60	1,2 ms	108°
Meñique	CCR3	60	1,2 ms	108°

3.8. Placa electrónica

Una vez detalladas las conexiones del ATmega328P (Figura 35) y del STM32 (Figura 43) se procede a unir estos diagramas en una sola placa electrónica la cual será diseñada en el Software “Proteus 8”, como se muestra en el Anexo 2 y que tendrá 7,1 cm de largo por 9 cm de altura (valores adquiridos tras medir el área disponible en la mano robótica).

Es importante mencionar que todas las entradas GND deben estar enlazadas para que el circuito funcione. Además, se implementará un voltaje de entrada que se encuentre entre 7V y 12V que es el rango soportado para alimentar directamente al ATmega328P como se explicó en el apartado teórico. Por otra parte, se colocará el convertidor MP1584EN en la entrada del circuito para que obtenga un voltaje de salida de 5V que energizará el STM32 y los servomotores (voltaje de entrada soportado según su manual) como se muestra en la Figura 61.

Adicionalmente, se procede a realizar cuatro conexiones con cables en ciertos puntos ya que las pistas pueden cruzarse y generar problemas, estos componentes se encuentran de color rojo y verde en el Anexo 2. Por último, se agregará un switch al inicio del circuito el cual controlará el encendido y apagado del sistema. El esquema electrónico elaborado en Proteus se lo puede apreciar de mejor manera en el Anexo 3.

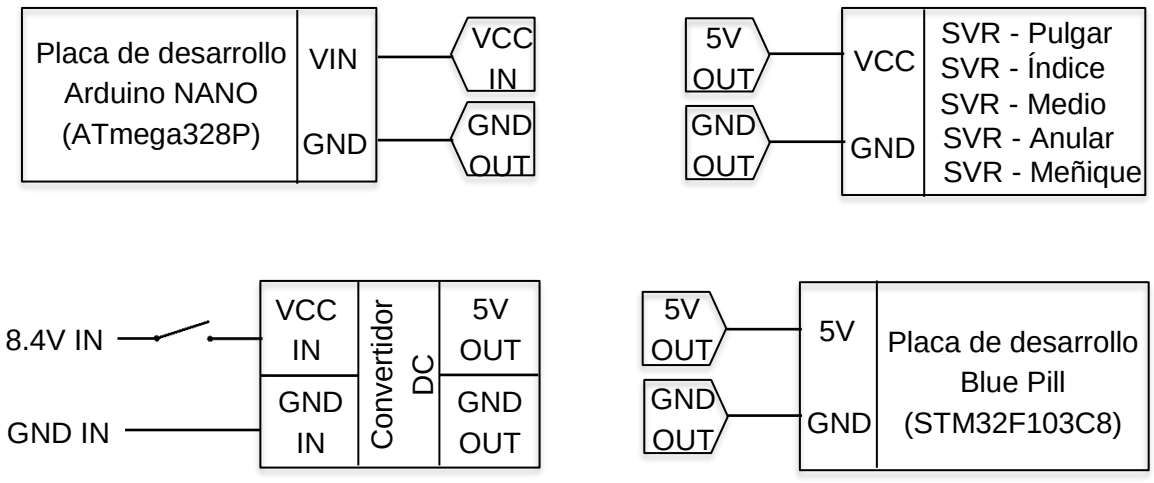


Figura 61. Circuito de alimentación con un voltaje de entrada de 8.4V.

Una vez ensamblada la placa electrónica y programado los microcontroladores se procede a realizar las pruebas respectivas y a documentar los resultados para su posterior análisis.

4. ANÁLISIS DE RESULTADOS

En este capítulo se describe y se analiza los resultados obtenidos tras la implementación del sistema de control de la mano robótica. Para ello, se tiene un grupo de once usuarios los cuales mantendrán abierta y cerrada la mano mientras usan el sensor MYO Armband en el área superior del antebrazo derecho.

Para las pruebas, se configurará al sistema microcontrolado para recolectar 25 ventanas electromiográficas de 30x8, 50x8, 70x8 y 100x8 (para cada movimiento) que serán procesadas, tomando las siguientes variables de medición:

- **Tiempo promedio de adquisición EMG (TPA):** Corresponde a la media de retardo (en milisegundos) que tiene el sistema desde el envío de la bandera por parte del STM32 hacia el ATmega328P, hasta la construcción de la ventana EMG en el procesador de 32 bits.
- **Tiempo promedio por reconocimiento (TPR):** Se refiere a la media del tiempo (en milisegundos) que se demora el STM32 en procesar la información, calcular las características y aplicarlas en el algoritmo clasificador Naïve Bayes para predecir el movimiento.
- **Tiempo total de prueba (TT):** Corresponde al tiempo que duró la toma y procesamiento de las 25 ventanas (en segundos).
- **Abierto:** Es el número de predicciones de mano abierta.
- **Cerrado:** Es el número de predicciones de mano cerrada.
- **No identificado (NI):** Es el número de casos en los que el sistema obtiene resultados menores al umbral (60%) en abierto y cerrado.

Para cada ventana electromiográfica se realiza un entrenamiento grupal que da como resultado una matriz (4x32) de constantes que son insertadas de forma manual en el sistema para que sean aplicadas por el algoritmo clasificador.

4.1. Entrenamiento del clasificador

El proceso de entrenamiento de Naïve Bayes es implementado en MATLAB, y de forma resumida, consiste en la toma de muestras EMG en un tiempo de diez segundos mientras se mantiene el movimiento, posteriormente se dimensiona la ventana y se extraen las características.

Finalmente se calcula los valores medios (MAV) y desviaciones estándar (STD) de los resultados que son exportados a un archivo Excel para ser ingresados en el sistema de forma manual.

Para una mejor comprensión de lo expuesto anteriormente, en la Figura 62 se presenta un esquema de la operación del algoritmo de entrenamiento.

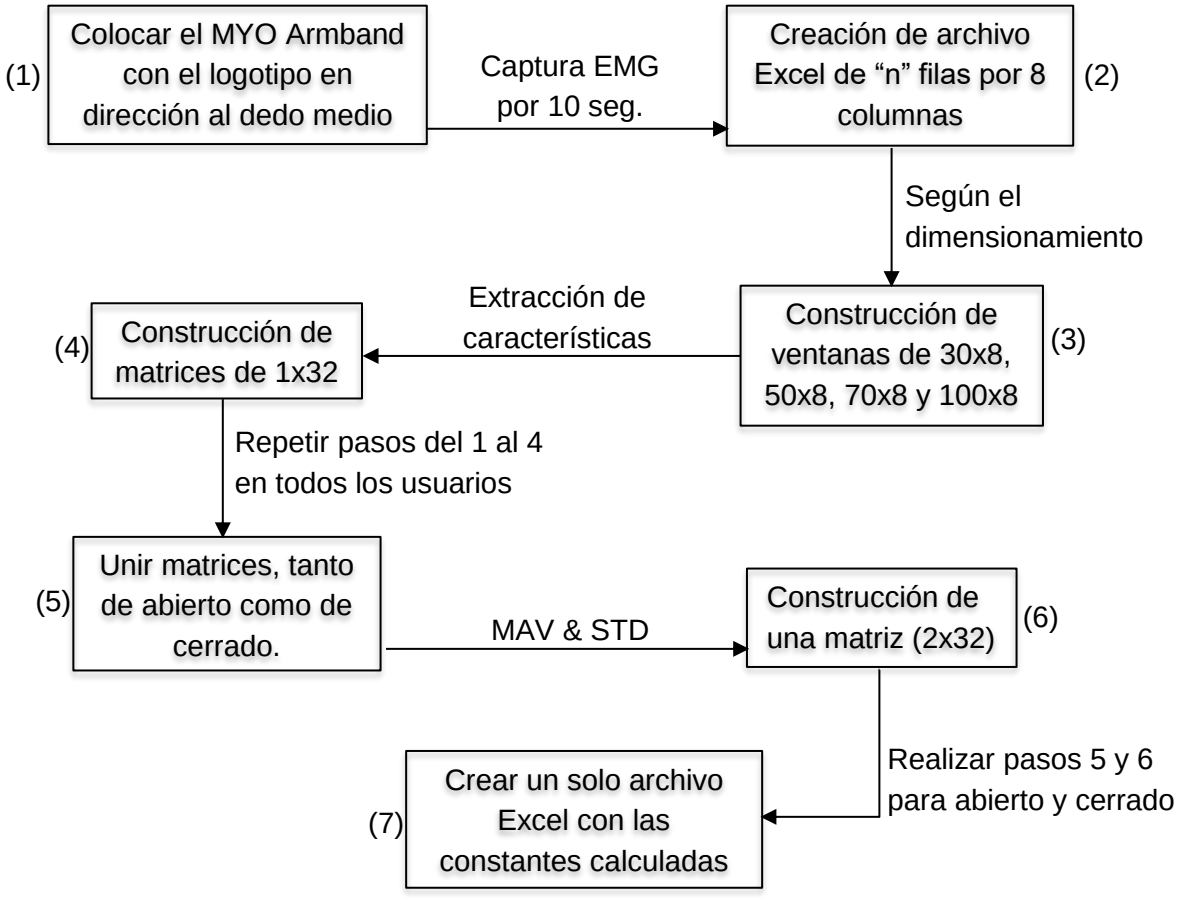


Figura 62. Algoritmo de entrenamiento de Naïve Bayes.

Los usuarios tomados para entrenar el algoritmo fueron elegidos según su género y contextura física con el objetivo de tener variedad en las señales recolectadas. Se formó un grupo de siete personas de las cuales cuatro son hombres y tres son mujeres.

4.2. Resultados de las pruebas

Las pruebas de operación consistieron en recolectar 25 ventanas mientras se mantiene un movimiento. Al finalizar el proceso, el microcontrolador procede a mostrar las variables de medición que serán registradas de forma manual para el posterior análisis de confusión y exactitud.

Las evaluaciones fueron aplicadas tanto a las siete personas entrenadas, como a cuatro personas independientes al sistema, con el objetivo de evaluar la adaptabilidad en usuarios nuevos.

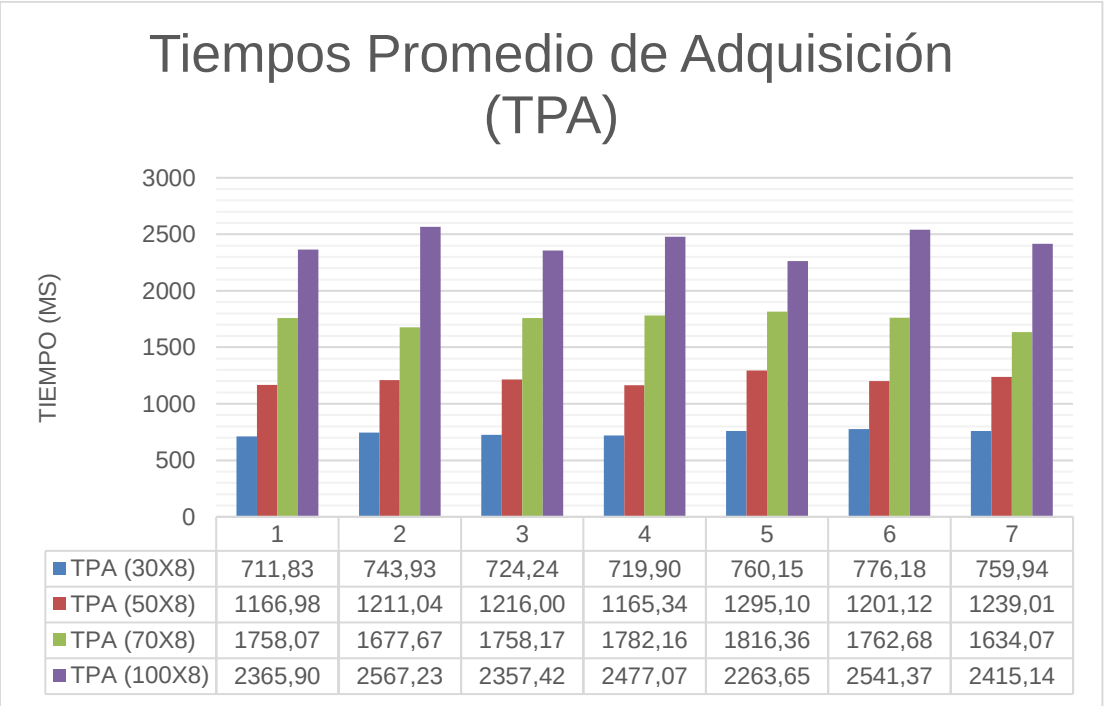
4.2.1. Pruebas con usuarios entrenados

En el siguiente apartado se muestran los resultados de las evaluaciones desarrolladas a las personas que realizaron el entrenamiento del clasificador. Los usuarios, además de realizar los dos movimientos propuestos, hicieron un tercero el cual consistía en colocar la mano en una posición neutral (ni abierta ni cerrada).

Esto se realizó para poder medir qué tanto se puede identificar un movimiento diferente a los configurados. Dicho esto, a continuación se procede a mostrar diversas gráficas representativas de los datos recolectados.

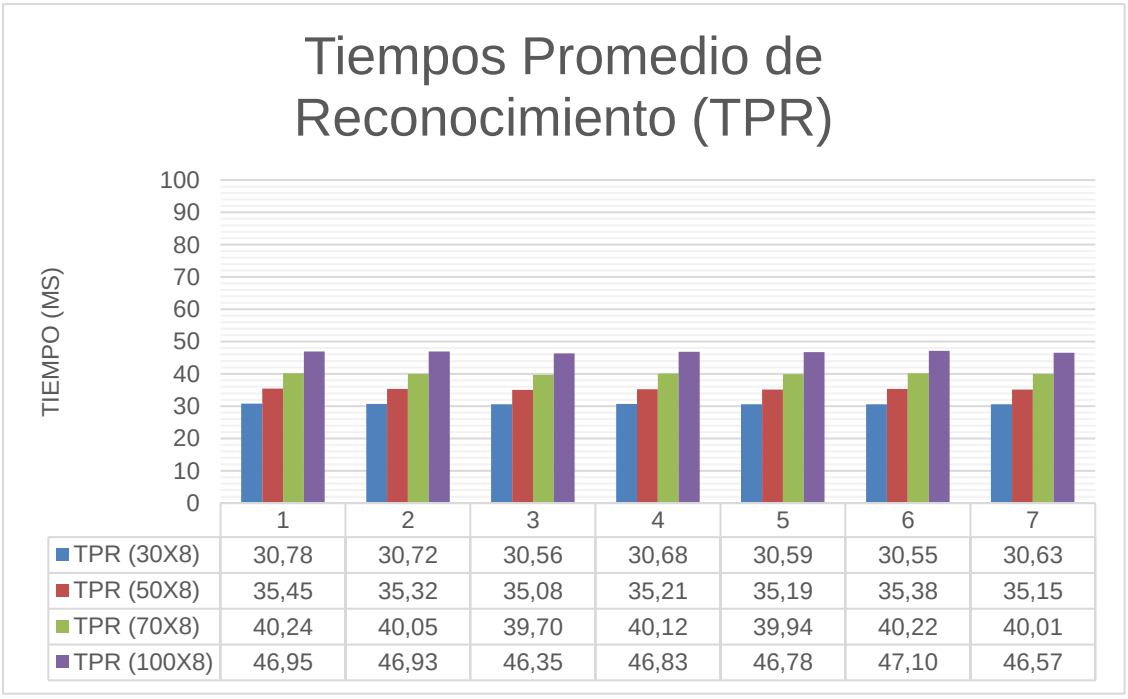
4.2.1.1. Mano abierta

En las Figuras 63, 64 y 65 se puede apreciar representación gráfica de las pruebas realizadas para mano abierta en ventanas de 30x8, 50x8, 70x8 y 100x8. En este caso se considera al movimiento abierto como acierto y cerrado como fallo. Las tablas que permitieron crear estas Figuras se las puede encontrar en el Anexo 4.



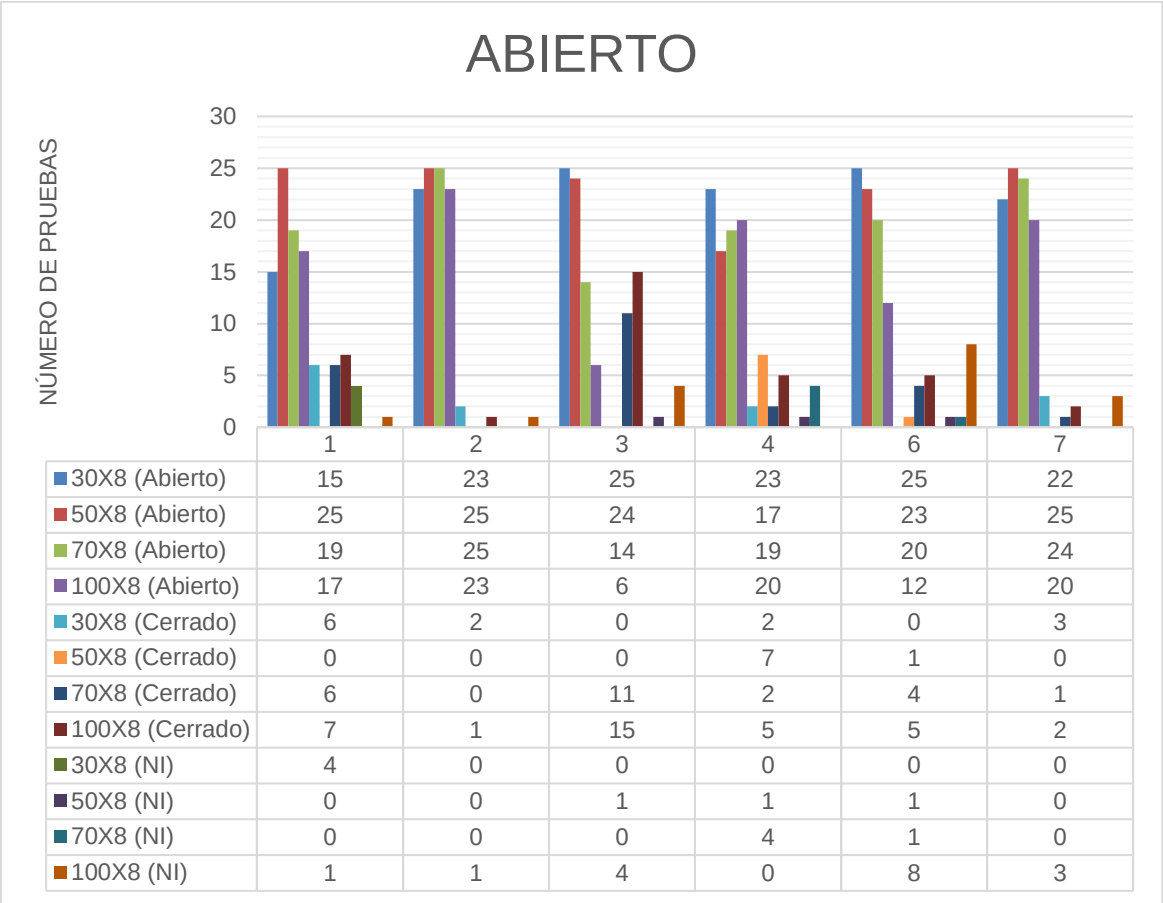
	1	2	3	4	5	6	7
TPA (30X8)	711,83	743,93	724,24	719,90	760,15	776,18	759,94
TPA (50X8)	1166,98	1211,04	1216,00	1165,34	1295,10	1201,12	1239,01
TPA (70X8)	1758,07	1677,67	1758,17	1782,16	1816,36	1762,68	1634,07
TPA (100X8)	2365,90	2567,23	2357,42	2477,07	2263,65	2541,37	2415,14

Figura 63. TPA con diferentes ventanas EMG para mano abierta.



	1	2	3	4	5	6	7
TPR (30X8)	30,78	30,72	30,56	30,68	30,59	30,55	30,63
TPR (50X8)	35,45	35,32	35,08	35,21	35,19	35,38	35,15
TPR (70X8)	40,24	40,05	39,70	40,12	39,94	40,22	40,01
TPR (100X8)	46,95	46,93	46,35	46,83	46,78	47,10	46,57

Figura 64. TPR con diferentes ventanas EMG para mano abierta.



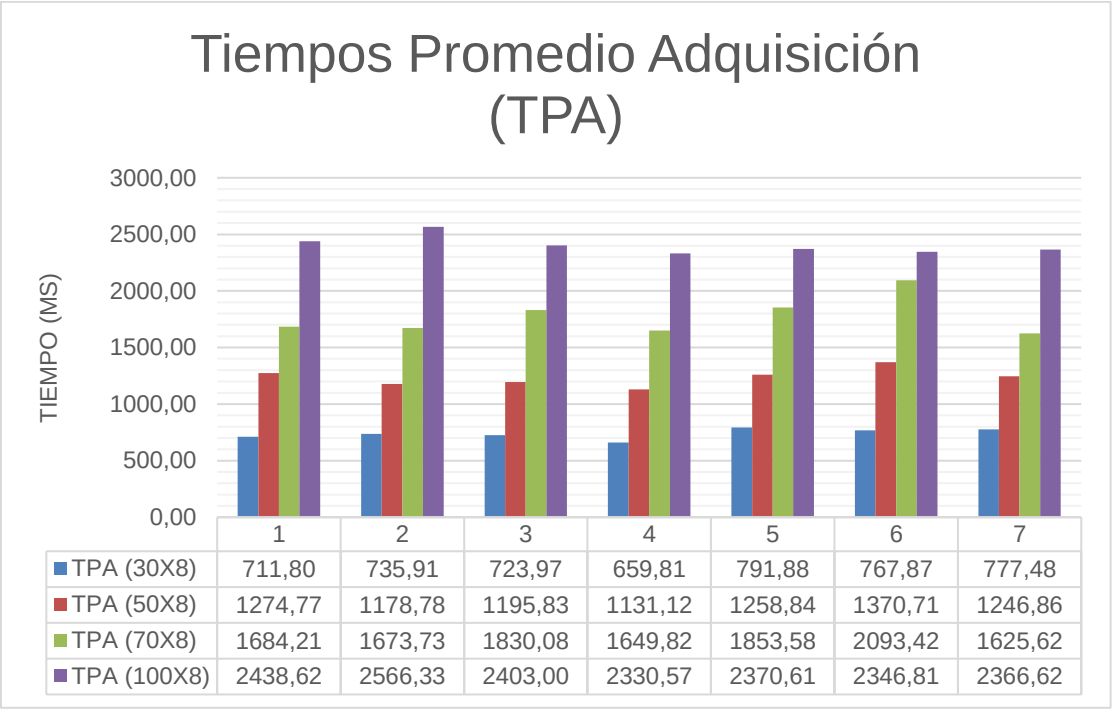
	1	2	3	4	6	7
30X8 (Abierto)	15	23	25	23	25	22
50X8 (Abierto)	25	25	24	17	23	25
70X8 (Abierto)	19	25	14	19	20	24
100X8 (Abierto)	17	23	6	20	12	20
30X8 (Cerrado)	6	2	0	2	0	3
50X8 (Cerrado)	0	0	0	7	1	0
70X8 (Cerrado)	6	0	11	2	4	1
100X8 (Cerrado)	7	1	15	5	5	2
30X8 (NI)	4	0	0	0	0	0
50X8 (NI)	0	0	1	1	1	0
70X8 (NI)	0	0	0	4	1	0
100X8 (NI)	1	1	4	0	8	3

Figura 65. Resultados de las pruebas realizadas para mano abierta.

En el gráfico expuesto en la Figura 65, se puede observar que con ventanas de 30X8 se ha reconocido el movimiento de mano abierta casi en su totalidad, ya que se obtiene valores de aciertos cercanos o iguales a 25 en casi todas las pruebas.

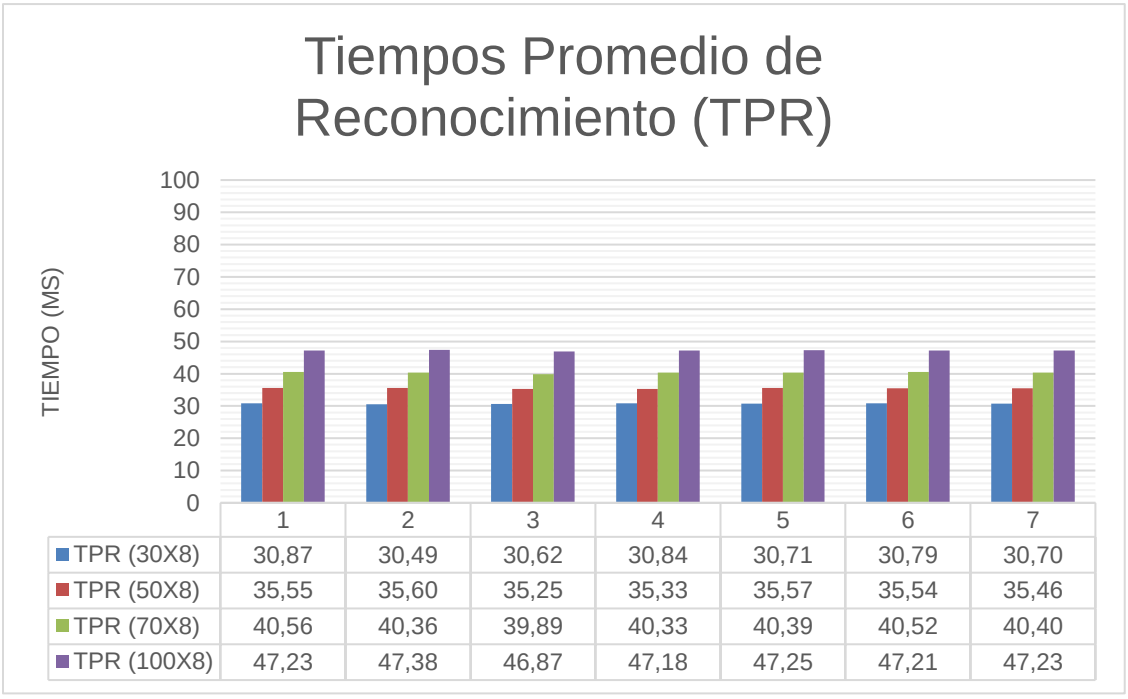
4.2.1.2. Mano cerrada

En las Figuras 66, 67 y 68 se puede apreciar la representación de las pruebas realizadas para mano cerrada ejecutadas igualmente con las ventanas planteadas. Para estas tabulaciones el movimiento cerrado es considerado como acierto, mientras que abierto es tomado como fallo. Las tablas que permitieron crear estas Figuras se las puede encontrar en el Anexo 5.



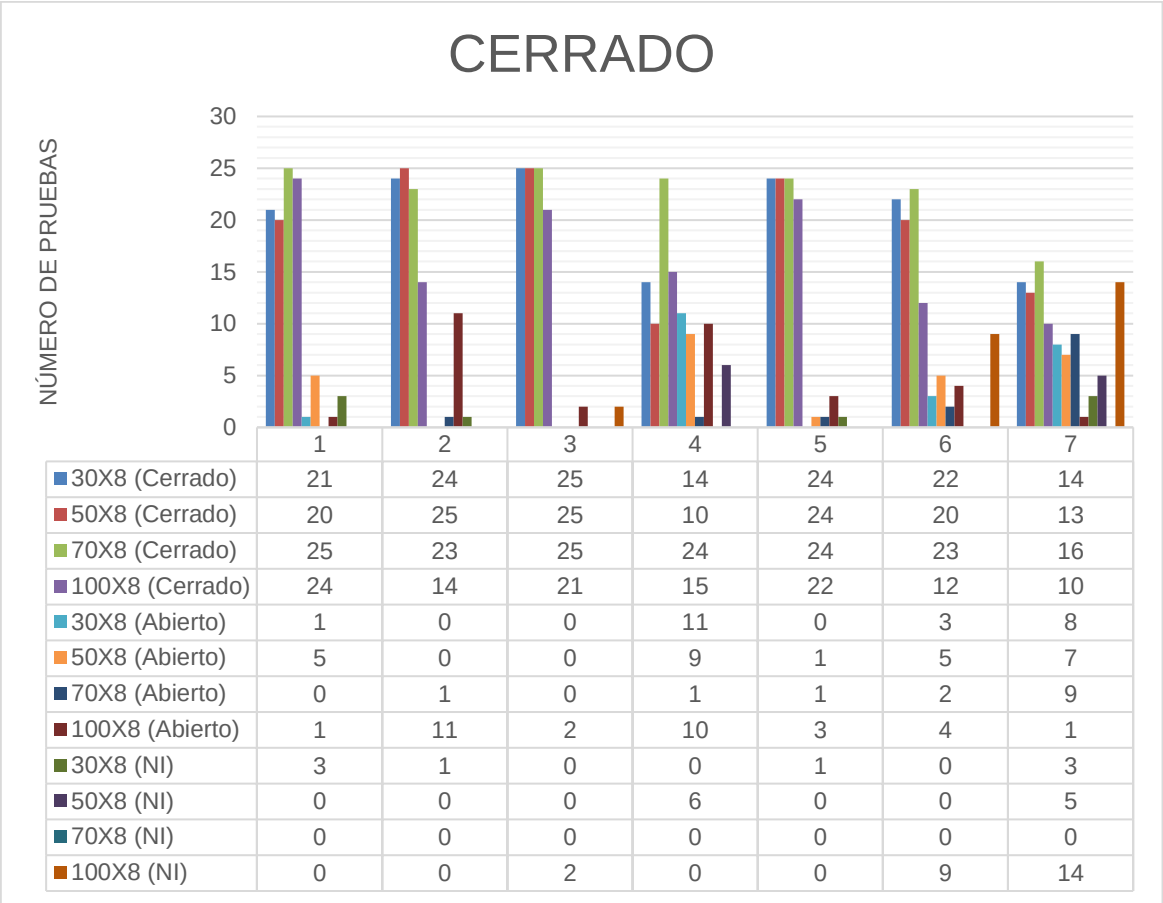
	1	2	3	4	5	6	7
TPA (30X8)	711,80	735,91	723,97	659,81	791,88	767,87	777,48
TPA (50X8)	1274,77	1178,78	1195,83	1131,12	1258,84	1370,71	1246,86
TPA (70X8)	1684,21	1673,73	1830,08	1649,82	1853,58	2093,42	1625,62
TPA (100X8)	2438,62	2566,33	2403,00	2330,57	2370,61	2346,81	2366,62

Figura 66. TPA con diferentes ventanas EMG para mano cerrada.



	1	2	3	4	5	6	7
TPR (30X8)	30,87	30,49	30,62	30,84	30,71	30,79	30,70
TPR (50X8)	35,55	35,60	35,25	35,33	35,57	35,54	35,46
TPR (70X8)	40,56	40,36	39,89	40,33	40,39	40,52	40,40
TPR (100X8)	47,23	47,38	46,87	47,18	47,25	47,21	47,23

Figura 67. TPR con diferentes ventanas EMG para mano cerrada.



	1	2	3	4	5	6	7
30X8 (Cerrado)	21	24	25	14	24	22	14
50X8 (Cerrado)	20	25	25	10	24	20	13
70X8 (Cerrado)	25	23	25	24	24	23	16
100X8 (Cerrado)	24	14	21	15	22	12	10
30X8 (Abierto)	1	0	0	11	0	3	8
50X8 (Abierto)	5	0	0	9	1	5	7
70X8 (Abierto)	0	1	0	1	1	2	9
100X8 (Abierto)	1	11	2	10	3	4	1
30X8 (NI)	3	1	0	0	1	0	3
50X8 (NI)	0	0	0	6	0	0	5
70X8 (NI)	0	0	0	0	0	0	0
100X8 (NI)	0	0	2	0	0	9	14

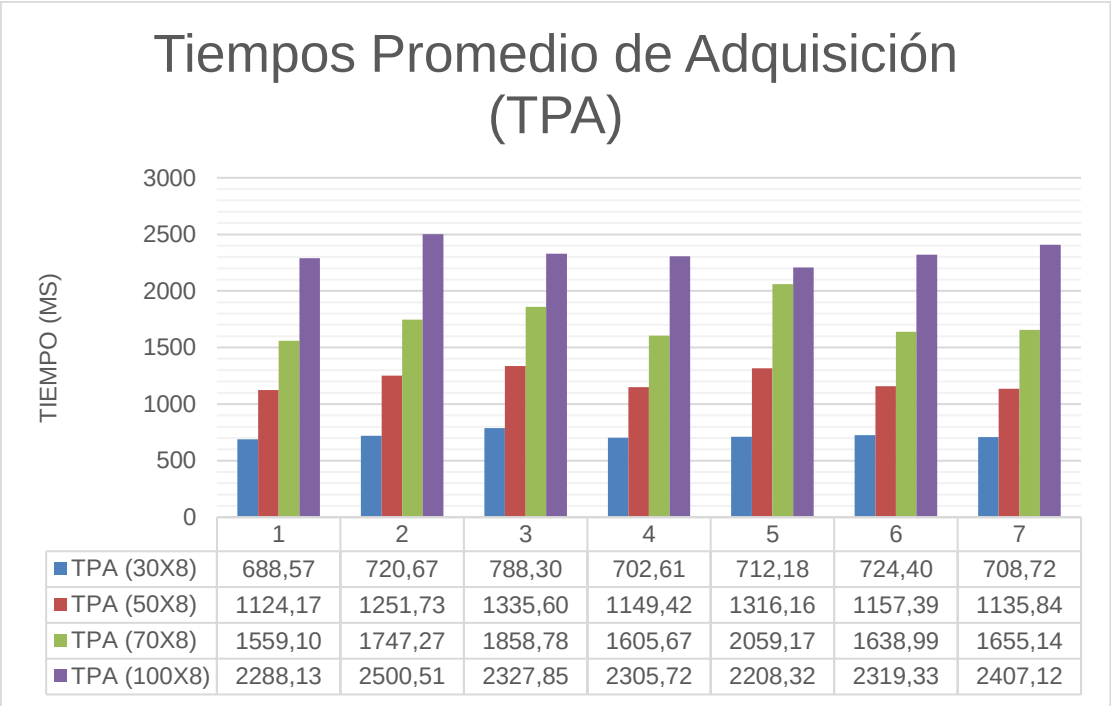
Figura 68. Resultados de las pruebas realizadas para mano cerrada.

4.2.1.3. Movimiento neutral

En las Figuras 69, 70 y 71 se puede apreciar representación gráfica de las pruebas realizadas para un movimiento no identificado (neutral) el cuál es registrado en la columna NI.

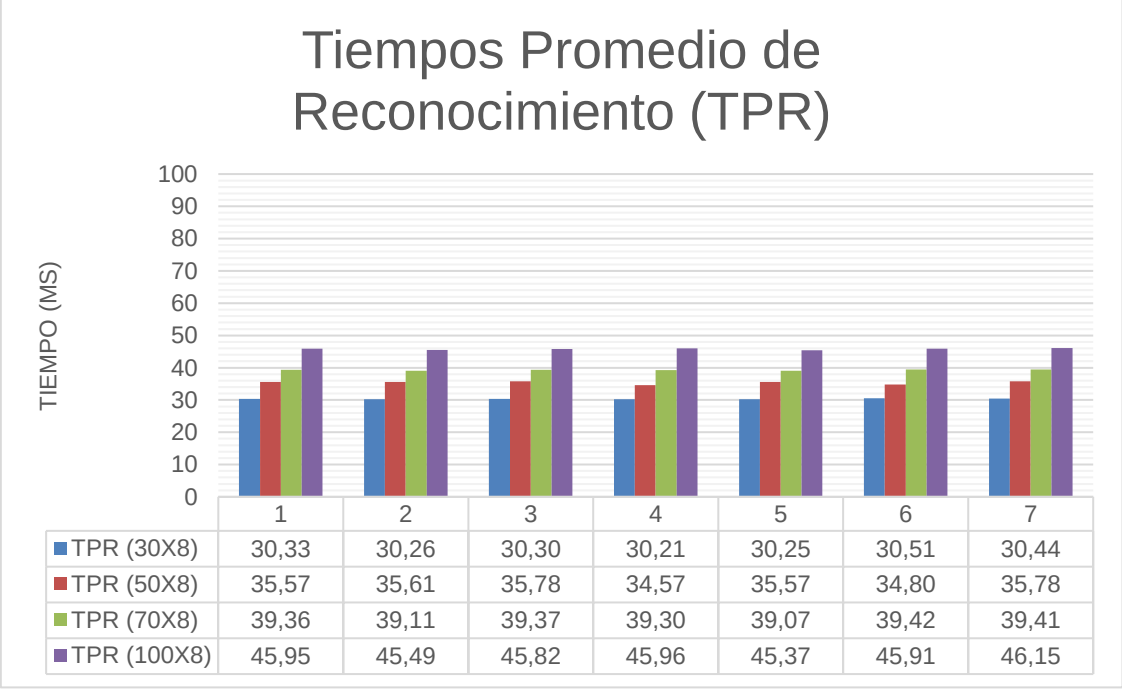
Básicamente, se registran las veces que el sistema acertó al no identificar el movimiento y las veces que se confundió con mano abierta o mano cerrada. Los movimientos abierto y cerrado son considerados como fallos.

Las tablas que permitieron crear estas Figuras se las puede encontrar en el Anexo 6.



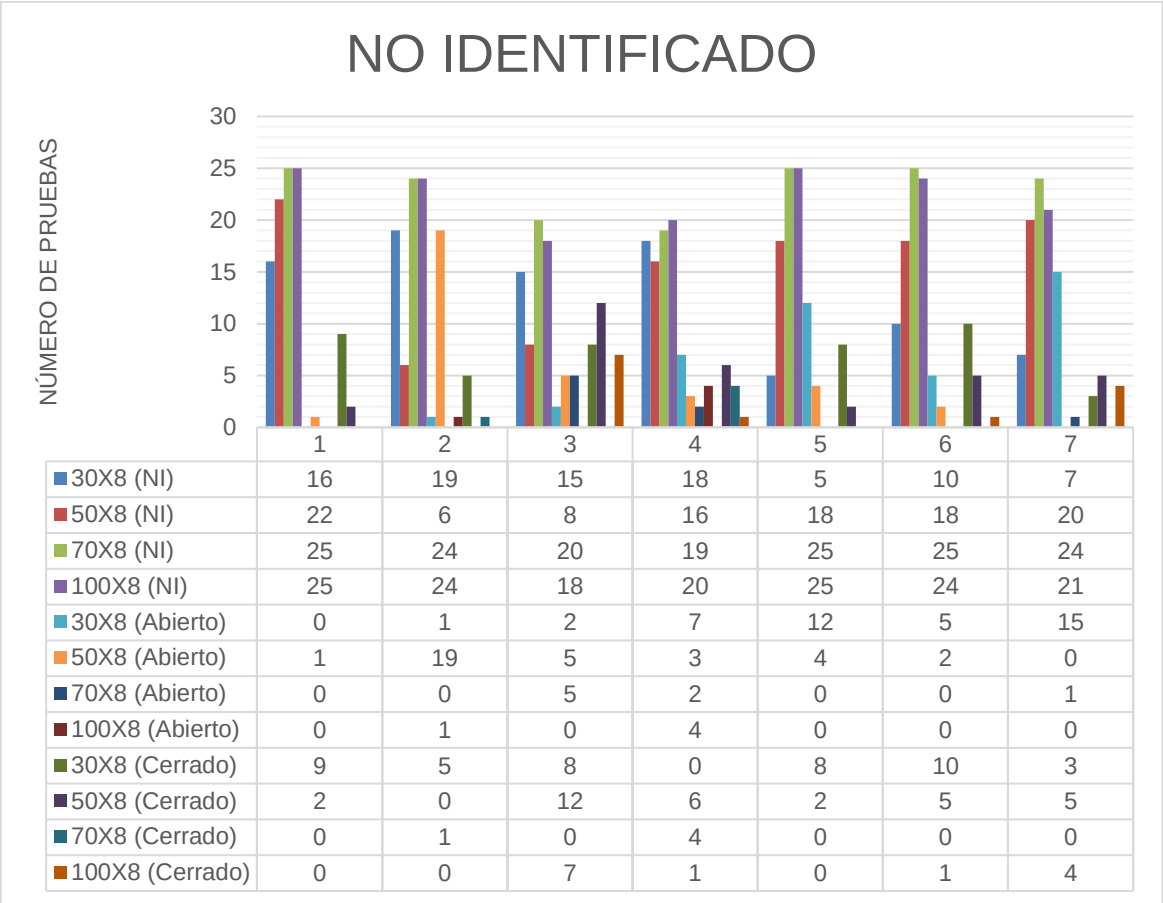
	1	2	3	4	5	6	7
TPA (30X8)	688,57	720,67	788,30	702,61	712,18	724,40	708,72
TPA (50X8)	1124,17	1251,73	1335,60	1149,42	1316,16	1157,39	1135,84
TPA (70X8)	1559,10	1747,27	1858,78	1605,67	2059,17	1638,99	1655,14
TPA (100X8)	2288,13	2500,51	2327,85	2305,72	2208,32	2319,33	2407,12

Figura 69. TPA con diferentes ventanas EMG para movimiento neutral (NI).



	1	2	3	4	5	6	7
TPR (30X8)	30,33	30,26	30,30	30,21	30,25	30,51	30,44
TPR (50X8)	35,57	35,61	35,78	34,57	35,57	34,80	35,78
TPR (70X8)	39,36	39,11	39,37	39,30	39,07	39,42	39,41
TPR (100X8)	45,95	45,49	45,82	45,96	45,37	45,91	46,15

Figura 70. TPR con diferentes ventanas EMG para movimiento neutral (NI).



	1	2	3	4	5	6	7
30X8 (NI)	16	19	15	18	5	10	7
50X8 (NI)	22	6	8	16	18	18	20
70X8 (NI)	25	24	20	19	25	25	24
100X8 (NI)	25	24	18	20	25	24	21
30X8 (Abierto)	0	1	2	7	12	5	15
50X8 (Abierto)	1	19	5	3	4	2	0
70X8 (Abierto)	0	0	5	2	0	0	1
100X8 (Abierto)	0	1	0	4	0	0	0
30X8 (Cerrado)	9	5	8	0	8	10	3
50X8 (Cerrado)	2	0	12	6	2	5	5
70X8 (Cerrado)	0	1	0	4	0	0	0
100X8 (Cerrado)	0	0	7	1	0	1	4

Figura 71. Resultados de las pruebas realizadas para movimiento neutral (NI).

4.2.2. Pruebas con usuarios no entrenados

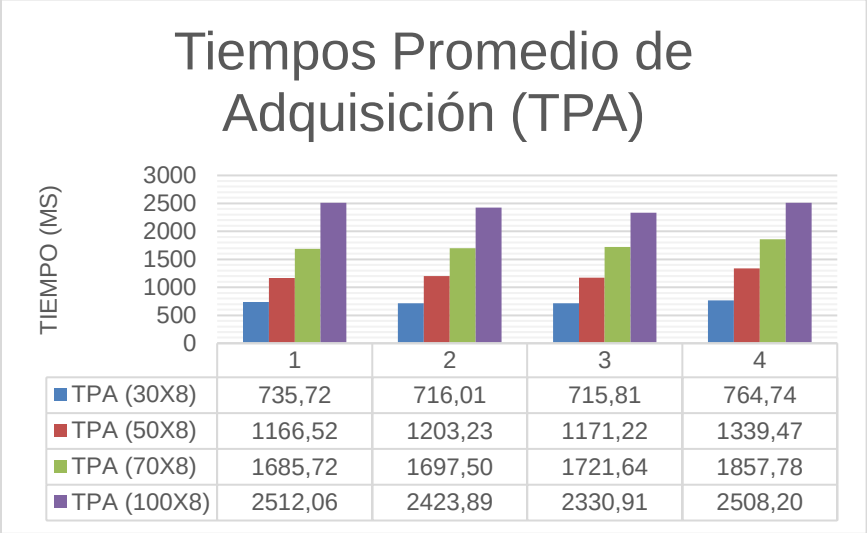
Una de las características que debe cumplir un algoritmo clasificador, es su capacidad de adaptarse correctamente en usuarios que no se encuentren registrados en su configuración.

Para demostrar que el sistema funciona con personas ajenas, se tomó a un grupo dos hombres y dos mujeres que realizaron las mismas pruebas planteadas anteriormente.

Dicho esto, a continuación se muestran los resultados de estas evaluaciones.

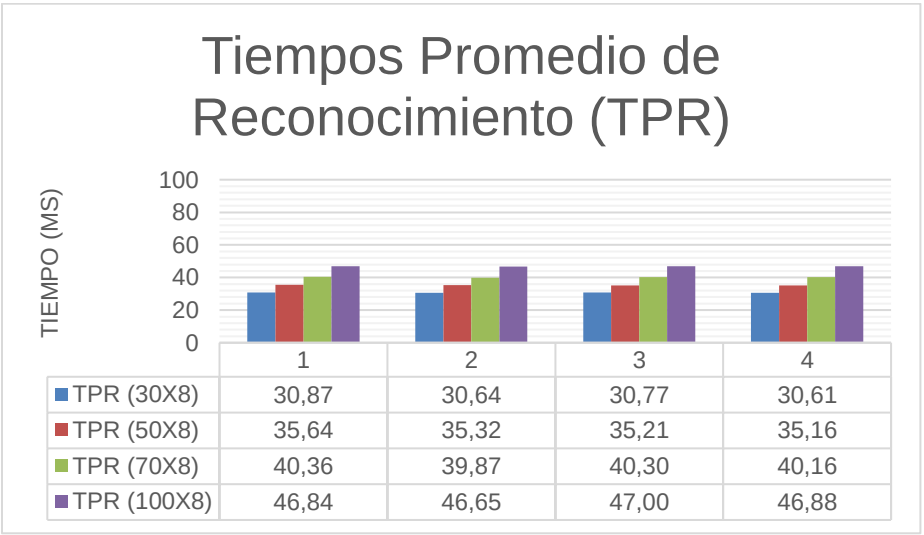
4.2.2.1. Mano abierta

En las Figuras 72, 73 y 74 se puede apreciar la información representada en gráficos de las pruebas realizadas para mano abierta en las diferentes ventanas. En estas pruebas se considera a abierto como acierto y cerrado como Fallo. Las tablas que permitieron crear estas Figuras se las puede encontrar en el Anexo 7.



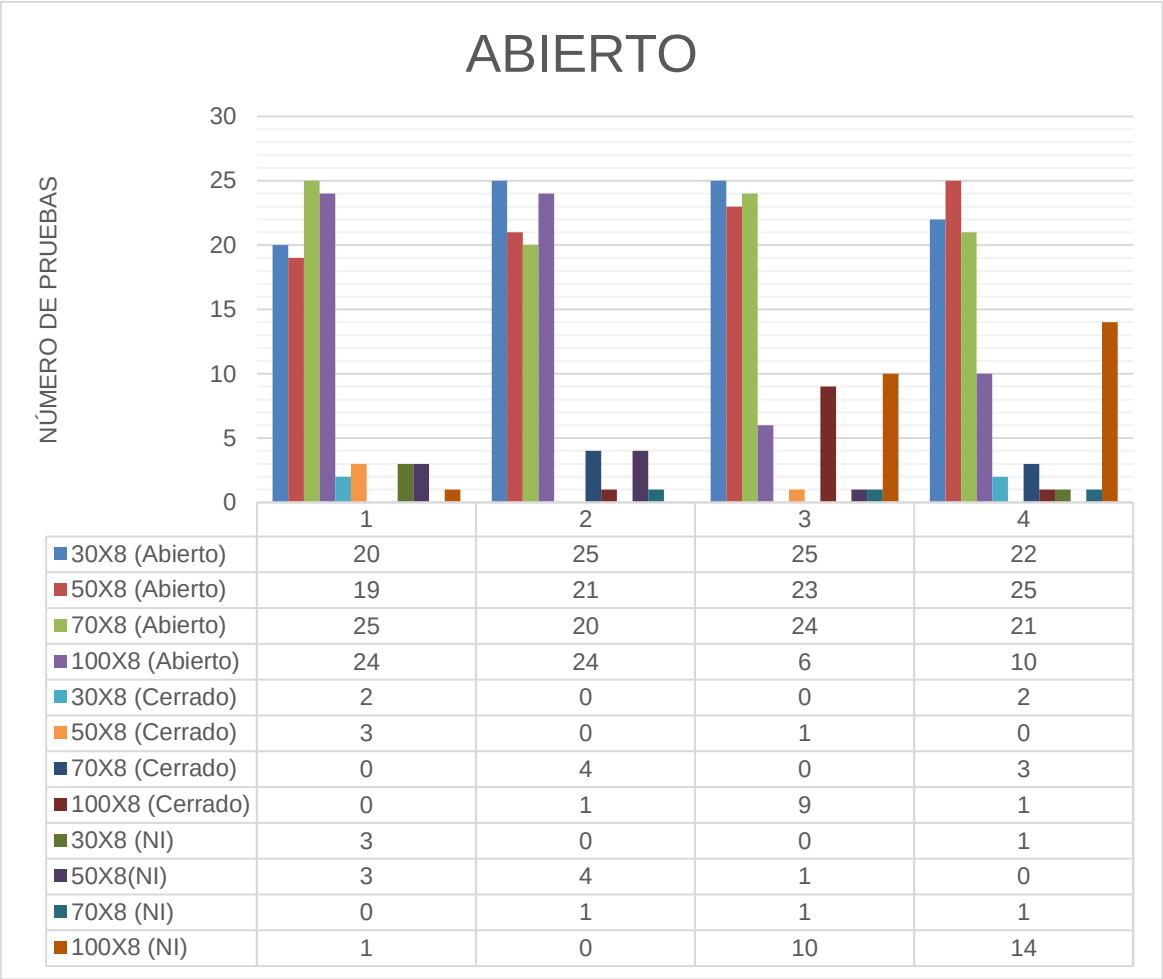
	1	2	3	4
TPA (30X8)	735,72	716,01	715,81	764,74
TPA (50X8)	1166,52	1203,23	1171,22	1339,47
TPA (70X8)	1685,72	1697,50	1721,64	1857,78
TPA (100X8)	2512,06	2423,89	2330,91	2508,20

Figura 72. TPA con diferentes ventanas EMG para mano abierta.



	1	2	3	4
TPR (30X8)	30,87	30,64	30,77	30,61
TPR (50X8)	35,64	35,32	35,21	35,16
TPR (70X8)	40,36	39,87	40,30	40,16
TPR (100X8)	46,84	46,65	47,00	46,88

Figura 73. TPR con diferentes ventanas EMG para mano abierta.



	1	2	3	4
30X8 (Abierto)	20	25	25	22
50X8 (Abierto)	19	21	23	25
70X8 (Abierto)	25	20	24	21
100X8 (Abierto)	24	24	6	10
30X8 (Cerrado)	2	0	0	2
50X8 (Cerrado)	3	0	1	0
70X8 (Cerrado)	0	4	0	3
100X8 (Cerrado)	0	1	9	1
30X8 (NI)	3	0	0	1
50X8(NI)	3	4	1	0
70X8 (NI)	0	1	1	1
100X8 (NI)	1	0	10	14

Figura 74. Resultados de las pruebas realizadas para mano abierta.

4.2.2.2. Mano cerrada

En las Figuras 75, 76 y 77 se puede observar los resultados tabulados de las pruebas para mano cerrada con las cuatro ventanas planteadas. Del mismo modo, en el movimiento cerrado se registran los aciertos y en abierto los fallos obtenidos en las pruebas.

Las tablas que permitieron crear estas Figuras se las puede encontrar en el Anexo 8.

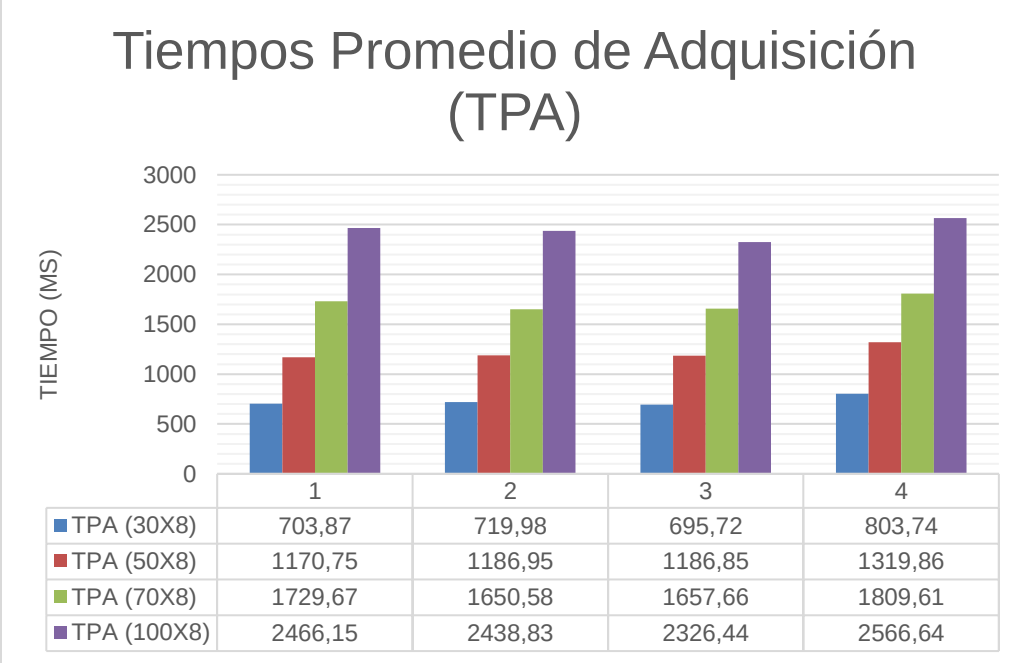


Figura 75. TPA con diferentes ventanas EMG para mano cerrada.

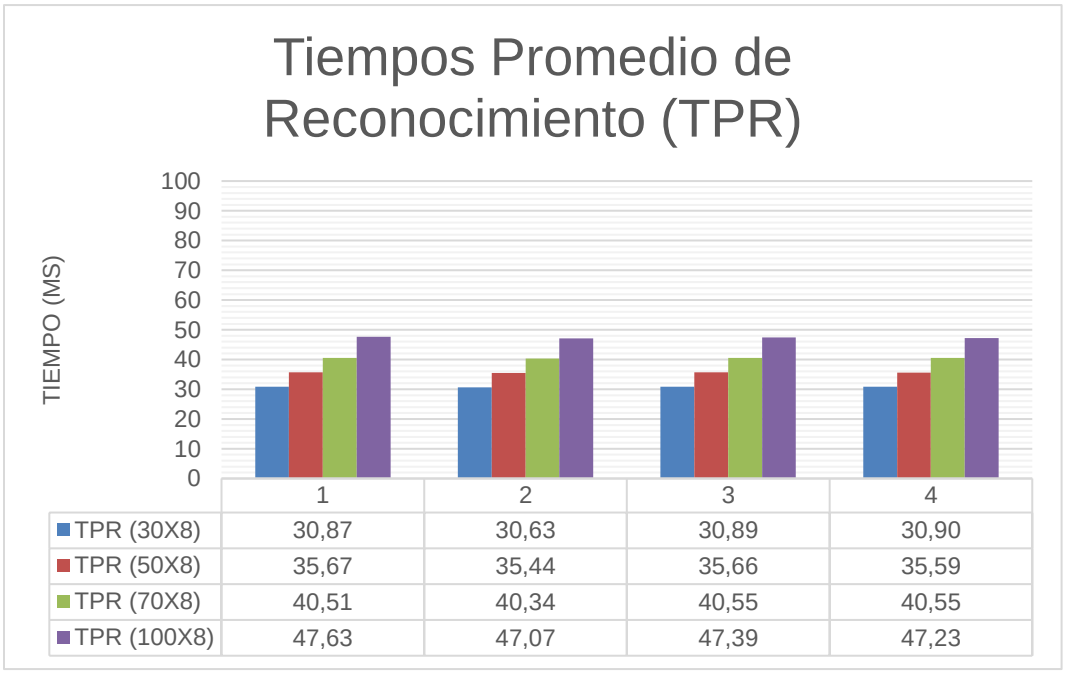
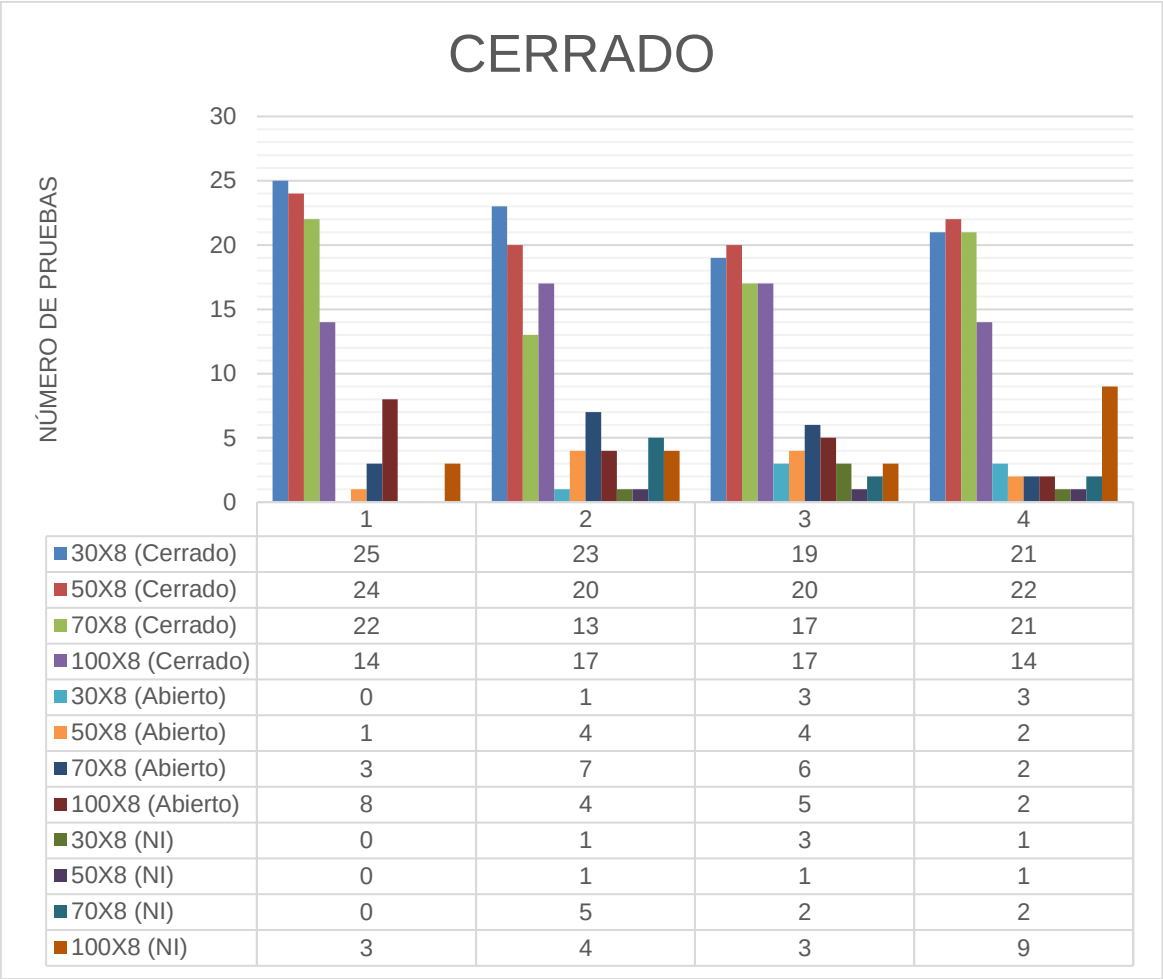


Figura 76. TPA con diferentes ventanas EMG para mano cerrada.



	1	2	3	4
30X8 (Cerrado)	25	23	19	21
50X8 (Cerrado)	24	20	20	22
70X8 (Cerrado)	22	13	17	21
100X8 (Cerrado)	14	17	17	14
30X8 (Abierto)	0	1	3	3
50X8 (Abierto)	1	4	4	2
70X8 (Abierto)	3	7	6	2
100X8 (Abierto)	8	4	5	2
30X8 (NI)	0	1	3	1
50X8 (NI)	0	1	1	1
70X8 (NI)	0	5	2	2
100X8 (NI)	3	4	3	9

Figura 77. Resultados de las pruebas realizadas para mano cerrada.

4.2.2.3. Movimiento neutral

En las Figuras 78, 79 y 80 se pueden apreciar los resultados obtenidos tras realizar las pruebas del movimiento no identificado (neutral) a los usuarios cuyos datos no forman parte del sistema.

Así mismo, en este caso sólo los registros de predicciones no identificadas son considerados como aciertos, por lo que los movimientos de abierto y cerrado son interpretadas como fallos en estas pruebas. Las tablas que permitieron crear estas Figuras se las puede encontrar en el Anexo 9.

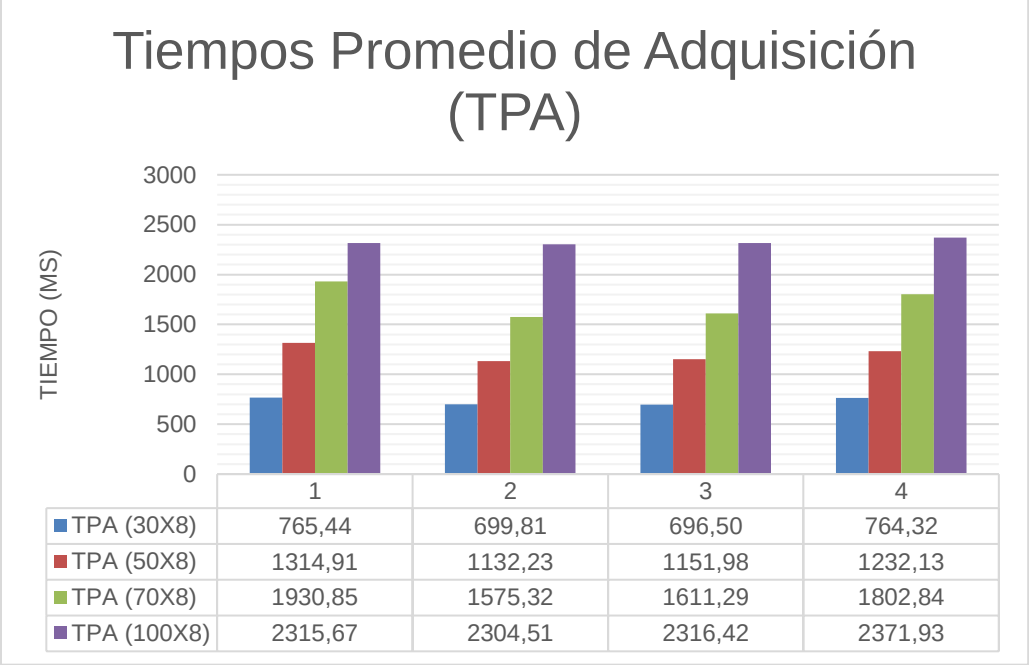


Figura 78. TPA con diferentes ventanas EMG para movimiento neutral (NI).

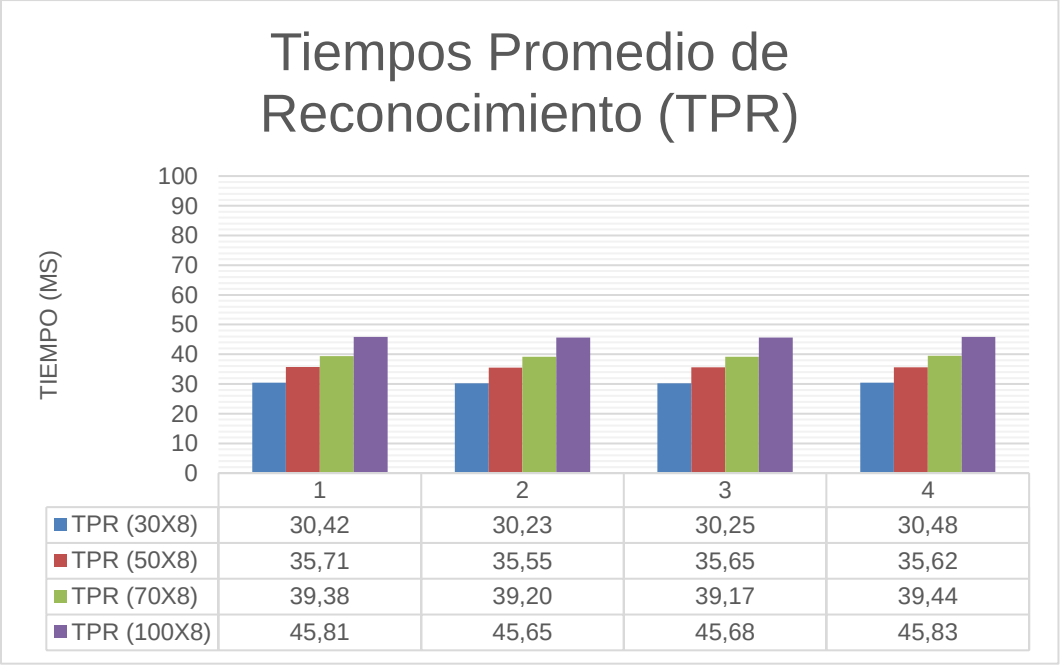
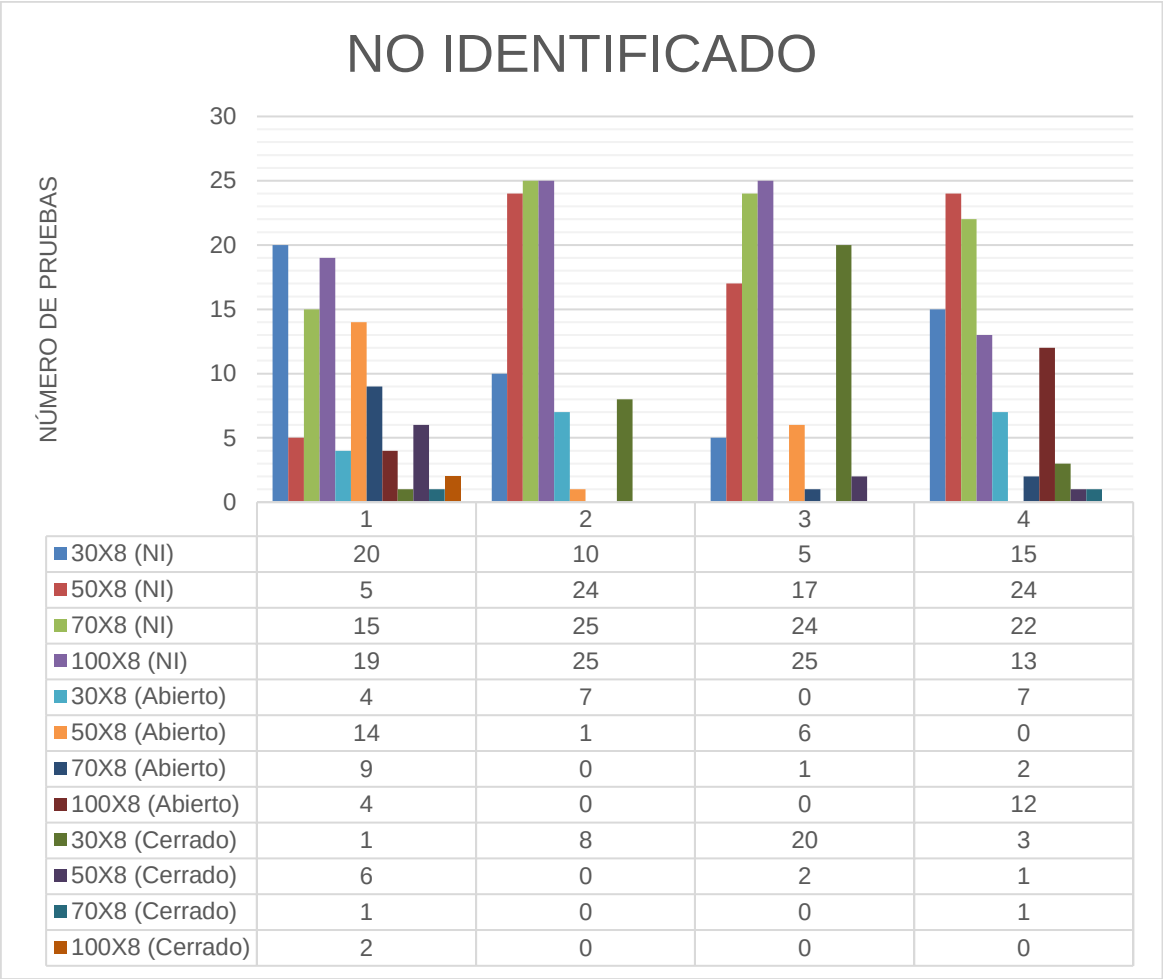


Figura 79. TPR con diferentes ventanas EMG para movimiento neutral (NI).



	1	2	3	4
30X8 (NI)	20	10	5	15
50X8 (NI)	5	24	17	24
70X8 (NI)	15	25	24	22
100X8 (NI)	19	25	25	13
30X8 (Abierto)	4	7	0	7
50X8 (Abierto)	14	1	6	0
70X8 (Abierto)	9	0	1	2
100X8 (Abierto)	4	0	0	12
30X8 (Cerrado)	1	8	20	3
50X8 (Cerrado)	6	0	2	1
70X8 (Cerrado)	1	0	0	1
100X8 (Cerrado)	2	0	0	0

Figura 80. Resultados de las pruebas realizadas para movimiento neutral (NI).

4.3. Evaluación de los resultados

En este apartado se procede a utilizar matrices de confusión, mismas que son un punto de partida para evaluar qué tan fiable y qué exactitud tiene un algoritmo de clasificación. Para lograr estas tablas, se hace un análisis entre el movimiento real o actual vs. la predicción realizada en las pruebas como se muestra en el modelo expuesto en la Tabla 6 que se utilizará para explicar el cuerpo de la matriz que se resume en la suma de abiertos, cerrados y no identificados obtenidos en las pruebas. Al finalizar, se debe tener una diagonal formada por los valores más altos de la matriz.

En este caso se utilizó letras para representar los valores de la tabla, a continuación se explica su representación:

- A: Predicciones correctas de mano abierta.
- B: Predicciones incorrectas de mano cerrada.
- C: Predicciones incorrectas de movimientos no identificados.
- D: Predicciones incorrectas de mano abierta.
- E: Predicciones correctas de mano cerrada.
- F: Predicciones incorrectas de movimientos no identificados.
- G: Predicciones incorrectas de mano abierta.
- H: Predicciones incorrectas de mano cerrada.
- I: Predicciones correctas de movimientos no identificados.

Posterior a obtener las matrices de confusión, se procede a calcular la exactitud del sistema por cada ventana EMG, para ello se divide la suma de la diagonal para la suma total de la matriz de confusión. En la Ecuación 11 (Data School, 2014) se puede apreciar un ejemplo de lo anteriormente dicho para entenderlo de mejor manera.

Tabla 6.

Modelo de la matriz de confusión.

Matriz de confusión		**Predicción**		
		Abierto	**Cerrado**	**No identificado**
Real	**Abierto**	A	B	C
	Cerrado	D	E	F
	No identificado	G	H	I

$$Exactitud = \frac{A + E + I}{A + B + C + D + E + F + G + H + I} \qquad \text{(Ecuación 11)}$$

Una vez comprendido el proceso de evaluación que será aplicado a los resultados, en seguida se presenta la Figura 81 que corresponde a los porcentajes de exactitud del sistema de los dos grupos, así como también el respectivo análisis individual de cada ventana EMG. En los Anexos 10, 11, 12 y 13 se encuentran las matrices de confusión que permitieron llegar a dichas exactitudes.

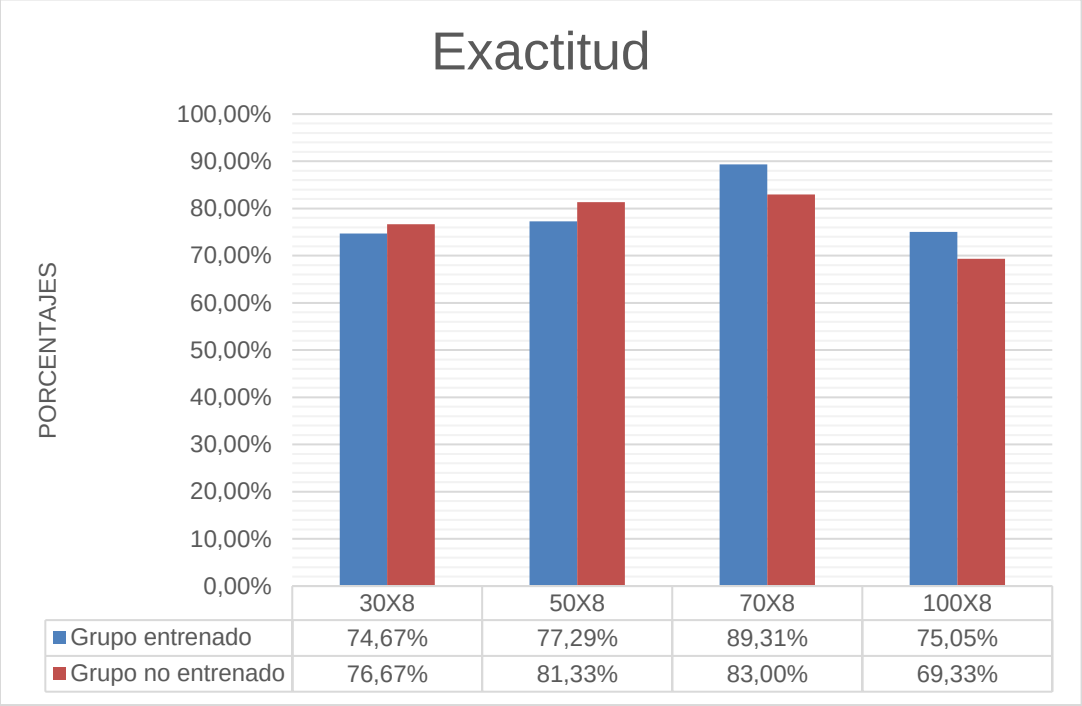


Figura 81. Exactitud del sistema por cada ventana EMG (grupo entrenado).

4.3.1. Ventana EMG de 30x8

Las pruebas realizadas con ventanas electromiográficas de 30x8 fueron las más rápidas en cuanto a tiempo de muestreo y tiempo de procesamiento según la Tabla 7 que indica los tiempos finales de la prueba.

Además, se obtuvo una exactitud del 74.67% en el reconocimiento de los gestos para el grupo entrenado y un 76.67% para el grupo no entrenado, lo cual permite concluir que con muestras de 30x8 se tiene una velocidad de procesamiento menor a 1 segundo, pero con cierta probabilidad de que existan fallos en las predicciones.

Finalmente, comparando los porcentajes de exactitud de la Figura 78, se puede observar que el algoritmo tuvo un 2% de mejor rendimiento en personas no entrenadas frente a personas entrenadas.

Tabla 7.

Resultados finales de las pruebas realizadas con ventanas EMG de 30x8.

Grupo	Tiempo promedio de muestreo (ms)	Tiempo promedio de procesamiento (ms)
Entrenado	733.83	30.56
No entrenado	731.81	30.63

4.3.2. Ventana EMG de 50x8

Las pruebas ejecutadas con ventanas electrográficas de 50x8, obtuvieron porcentajes de exactitud superiores con un 77.29% para el grupo entrenado y un 81.33% para el grupo no entrenado como se puede apreciar en la Figura 78.

Sin embargo el tiempo de respuesta se encuentra cerca de los 1.2 segundos, debido a que el sensor debe capturar y transmitir casi el doble de señales EMG en comparación a las primeras pruebas, como se observa en la Tabla 8.

Se concluye que la utilización de ventanas de 50x8 no muestran un cambio significativo en la exactitud ya que solo es un 2.62% mayor en usuarios entrenados y un 4.66% mayor en usuarios no entrenados, respecto a la utilización de ventanas de 30x8.

Tabla 8.

Resultados finales de las pruebas realizadas con ventanas EMG de 50x8.

Grupo	Tiempo promedio de muestreo (ms)	Tiempo promedio de procesamiento (ms)
Entrenado	1220.09	35.37
No entrenado	1214.68	31.25

4.3.3. Ventana EMG de 70x8

Con un 89.31% para usuarios entrenados y un 83% para usuarios no entrenados, las pruebas realizadas con ventanas electromiográficas de 70x8 fueron aquellas que resultaron tener el mayor porcentaje de exactitud de entre todas, como se puede apreciar en la Figura 78. Para el grupo que forma parte del sistema se tiene un aumento del 14.64% de exactitud mientras que para el otro grupo se aumentó un 6.33% frente a las ventanas de 30x8. Sin embargo, al tomar una porción EMG de gran tamaño, el tiempo de respuesta ha aumentado alrededor de 1,8 segundos por predicción (casi el doble de tiempo que toma la predicción con ventanas de 30x8). Así se concluye que, con muestras de 70x8 se tiene una alta exactitud y un tiempo de respuesta que supera el segundo, según la tabla 9.

Tabla 9.

Resultados finales de las pruebas realizadas con ventanas EMG de 70x8.

Grupo	Tiempo promedio de muestreo (ms)	Tiempo promedio de procesamiento (ms)
Entrenado	1748.75	39.89
No entrenado	1727.54	39.99

4.3.4. Ventana EMG de 100x8

Según los resultados mostrados en la Figura 78 , con una exactitud del 75.05% para el grupo de usuarios incluidos en el sistema y con un 69.33% para usuarios que no formaron parte del entrenamiento, las pruebas realizadas con ventanas electromiográficas de 100x8 fueron aquellas que dieron la más baja exactitud para ambos grupos mientras que también presentaron tiempos de respuesta demasiado largos, como se puede observar en la Tabla 10.

Se dice esto ya que con el grupo entrenado se tiene solamente un 0.38% menos de exactitud en comparación a las pruebas de 30x8, y poseen una diferencia significativa de 7.34% para el grupo no entrenado. Frente a las pruebas con ventanas de 50x8, estas exactitudes son menores por 2.4% en usuarios entrenados y 12% en usuarios no entrenados.

La diferencia más notable se la puede observar cuando se compara las exactitudes de esta ventana con las de 70x8, ya que estas son menores por 14.26% cuando se aplica en personas que fueron entrenadas y 13.67% en no entrenadas. En cuanto a tiempos, esta ventana ha proporcionado con un aproximado de 2.4 segundos de reconocimiento y 46.64 milisegundos de procesamiento, lo cual puede ser considerado como poco eficiente tanto en predicción como tiempo de respuesta.

Tabla 10.

Resultados finales de las pruebas realizadas con ventanas EMG de 100x8.

Grupo	Tiempo promedio de muestreo (ms)	Tiempo promedio de procesamiento (ms)
Entrenado	2388.92	46.60
No entrenado	2406.80	46.64

5. CONCLUSIONES Y RECOMENDACIONES

En este capítulo se expone las conclusiones y recomendaciones relevantes del trabajo de titulación.

5.1. Conclusiones

La experimentación desarrollada permite concluir que el rango electromiográfico oportuno para obtener buenos resultados son con muestras desde 30x8 hasta 70x8. La configuración de ventanas que se encuentren fuera de este rango puede ocasionar tiempos de respuesta extensos, porcentajes de exactitud demasiado bajos y mayor consumo de recursos del sistema microcontrolado.

A pesar de que ningún algoritmo clasificador es preciso en un 100%, la implementación de estos en el área de machine learning es fundamental para alcanzar objetivos y conseguir resultados cercanos al éxito. La utilización de Naïve Bayes en este trabajo de titulación ha permitido obtener exactitudes desde 75% hasta 89% con tiempos de predicción menores a los 65 milisegundos.

Una de las propiedades primordiales que se tiene en cuenta en un algoritmo de machine learning es su comportamiento frente a casos o usuarios que no se encuentren registrados dentro de su configuración. En las experimentaciones realizadas se propuso evaluar esto, consiguiendo una buena adaptabilidad del sistema y alcanzando porcentajes de predicción aceptables que permiten concluir que el sistema puede ser utilizado por cualquier usuario.

Los microcontroladores de 32 bits pueden utilizarse para desarrollar sistemas embebidos de machine learning. Se dice esto ya que los algoritmos ocupan una alta cantidad de recursos y procesamiento debido a que trabajan con grandes bloques de información que llegan de forma constante y que requieren ser procesados rápidamente.

5.2. Recomendaciones

Cuando se está desarrollando un sistema de machine learning, es recomendable construirlo con un alto nivel de escalabilidad tanto en la programación como en el diseño físico, con la finalidad de que en un futuro se pueda adicionar funciones que mejoren el prototipo. En este caso, aumentar la cantidad de movimientos configurados que se puedan predecir.

Al momento de utilizar el sistema de control de la mano robótica, es recomendable que el usuario se coloque el sensor MYO Armband en la posición correcta la cual es en la parte superior del antebrazo derecho con el logotipo apuntando hacia el dedo medio en línea recta. La mala posición del MYO Armband puede influenciar en los resultados de predicción.

Es recomendable siempre limpiar los electrodos del sensor MYO Armband con alcohol antiséptico y una gasa cuando este sea utilizado por varios usuarios con el objetivo de que no existan irritaciones en la piel en las personas que lo compartan. Además, se recomienda esperar unos segundos hasta que el sensor se adapte al calor corporal del usuario y comience a enviar señales EMG correctas.

Cuando se está inicializando el sistema microcontrolado, es recomendable esperar hasta que el LED indicador verde se encienda para presionar el botón RESET del STM32. Se dice esto ya que cuando se enciende el LED significa que el sensor MYO Armband se enlazó correctamente al sistema y puede comenzar a transmitir las señales electromiográficas.

REFERENCIAS

101 Components. (2018). Arduino Nano. Recuperado el 21 de abril de 2019, de https://components101.com/microcontrollers/arduino-nano

Active 8 Robots. (2019). Introducing the new AR10 Humanoid Robot Hand. Recuperado el 2 de abril de 2019, de https://www.active8robots.com/robots/ar10-robotic-hand/

Advancer Technologies. (2019). MyoWare Muscle Sensor. Recuperado el 11 de abril de 2019, de http://www.advancertechnologies.com/p/myoware.html

Aggarwal, C. C. (2014). Data Classification: Algorithms and Applications. Boca Raton: CRC Press. https://doi.org/10.1201/b17320

Alsharif, A. (2017). Connecting an Arduino to a Myo armband using a HM-10 / HM-11. Recuperado el 26 de abril de 2019, de https://blog.raquenaengineering.com/arduino-and-the-myo-armband/

Arduino. (2019a). ARDUINO NANO. Recuperado el 21 de abril de 2019, de https://store.arduino.cc/usa/arduino-nano

Arduino. (2019b). available(). Recuperado el 2 de mayo de 2019, de https://www.arduino.cc/en/Serial/Available

Arduino. (2019c). Begin. Recuperado el 1 de mayo de 2019, de https://www.arduino.cc/en/Serial/Begin

Arduino. (2019d). read(). Recuperado de https://www.arduino.cc/en/serial/read

Arduino. (2019e). Serial. Recuperado el 1 de mayo de 2019, de https://www.arduino.cc/en/pmwiki.php?n=Reference/Serial

Arduino. (2019f). SoftwareSerial. Recuperado el 1 de mayo de 2019, de

https://www.arduino.cc/en/Reference/SoftwareSerial

Arduino. (2019g). write(). Recuperado el 2 de mayo de 2019, de https://www.arduino.cc/en/Serial/Write

Arief, Z., Sulistijono, I. A., & Ardiansyah, R. A. (2015). Comparison of five time series EMG features extractions using Myo Armband. Proceedings - 2015 International Electronics Symposium: Emerging Technology in Electronic and Information, IES 2015, 11–14. https://doi.org/10.1109/ELECSYM.2015.7380805

ARM Keil. (2019). µVision IDE. Recuperado el 2 de mayo de 2019, de http://www2.keil.com/mdk5/uvision/

Barrios, J. (2014). Implementación de una Aplicación Informática de Análisis Biomecánico y Control Motor para el Entrenamiento de las Cualidades de la Fuerza Muscular. Universidad de Extremadura. https://doi.org/10.1177/2042098618776598

Bitalino. (2019). Electromyography (EMG) Sensor. Recuperado el 12 de abril de 2019, de https://store.plux.info/barebone-sensors/8-electromyography-emg-sensor.html

Chen, C. H., & Subbaram Naidu, D. (2013). Hybrid control strategies for smart robotic hand. Hybrid Control Strategies for Smart Robotic Hand, 8(4), 1–256. https://doi.org/10.1002/9781119273622

Cheng, H., Yang, L., & Liu, Z. (2015). Survey on 3D Hand Gesture Recognition. IEEE Transactions on Circuits and Systems for Video Technology, 26(9), 1659–1673. https://doi.org/10.1109/TCSVT.2015.2469551

Cheung, P. (2018). Servo Motor SG90. Micro motor, 1(2), 180. Recuperado de

http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

Controzzi, M., Cipriani, C., & Carrozza, M. C. (2014). Design of Artificial Hands: A Review. En R. Balasubramanian & V. J. Santos (Eds.), The Human Hand as an Inspiration for Robot Hand Development (95a ed., pp. 219–246). Suiza: Springer. https://doi.org/10.1007/978-3-319-03017-3_11

Correa-Figueroa, J. L., Morales-Sánchez, E., Huerta-Ruelas, J. A., González-Barbosa, J. J., & Cárdenas-Pérez, C. R. (2016). Sistema de adquisición de señales SEMG para la detección de fatiga muscular. Revista Mexicana de Ingenieria Biomedica, 37(1), 17–27. https://doi.org/10.17488/RMIB.37.1.4

Custom Entertainment Solutions. (2018). Mecha X Robotic Hand. Recuperado el 9 de noviembre de 2018, de https://animatronicrobotics.com/product/mecha-x-robotic-hand/

Custom Entertainment Solutions. (2019). DYN Hand GEN3. Recuperado el 2 de abril de 2019, de https://animatronicrobotics.com/product/dyn-hand-gen3/

Data School. (2014). Simple guide to confusion matrix terminology. Recuperado el 31 de mayo de 2019, de https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

Deal Extreme. (2019). ZHAOYAO CP2102 Módulo USB A TTL Serie UART STC Cable De Descarga PL2303 Súper Actualización De Línea De Cepillo - Rojo. Recuperado el 21 de abril de 2019, de https://www.dx.com/es/p/zhaoyao-cp2102-module-usb-to-ttl-serial-uart-stc-download-cable-pl2303-super-brush-line-upgrade-red-2089180#

DFRobot. (2019a). Bionic Robot Hand (Right). Recuperado el 2 de abril de 2019, de

https://www.dfrobot.com/wiki/index.php/Bionic_Robot_Hand_SKU:_ROB0142_%26_ROB0143

DFRobot. (2019b). Gravity: Analog EMG Sensor by OYMotion. Recuperado el 12 de abril de 2019, de https://www.dfrobot.com/product-1661.html

DSD TECH. (2018). DSD TECH HM-11 Bluetooth 4.0 BLE Module. Recuperado el 21 de abril de 2019, de http://www.dsdtech-global.com/2018/04/dsd-tech-hm-11.html

ElectroTools. (2016). COMO USAR LAS INTERRUPCIONES EN ARDUINO. Recuperado el 2 de mayo de 2019, de https://www.electrontools.com/Home/WP/2016/05/13/como-usar-las-interrupciones-en-arduino/

Embedded. (2016). DISCOVERY: CUBEMX. Recuperado el 6 de mayo de 2019, de https://www.embedded.fm/blog/2016/9/25/cubemx

Flood, M. W., Jensen, B. R., Malling, A. S., & Lowery, M. M. (2018). Increased EMG intermuscular coherence and reduced signal complexity in Parkinson's disease. Clinical Neurophysiology, 130(2), 259–269. https://doi.org/10.1016/j.clinph.2018.10.023

Ghassemi, M., Triandafilou, K., Barry, A., Stoykov, M. E., Roth, E., Mussa-Ivaldi, F. A., … Ranganathan, R. (2019). Development of an EMG-controlled Serious Game for Rehabilitation. IEEE Transactions on Neural Systems and Rehabilitation Engineering, PP(c), 1. https://doi.org/10.1109/TNSRE.2019.2894102

Gokgoz, E., & Subasi, A. (2015). Comparison of decision tree algorithms for EMG signal classification using DWT. Biomedical Signal Processing and Control, 18, 138–144. https://doi.org/10.1016/j.bspc.2014.12.005

Gruebler, A., & Suzuki, K. (2014). Design of a wearable device for reading positive expressions from facial EMG signals. IEEE Transactions on Affective Computing, 5(3), 227–237. https://doi.org/10.1109/TAFFC.2014.2313557

Health Line. (2015a). Brachioradialis. Recuperado el 17 de abril de 2019, de https://www.healthline.com/human-body-maps/brachioradialis-muscle#1

Health Line. (2015b). Extensor carpi ulnaris. Recuperado el 17 de abril de 2019, de https://www.healthline.com/human-body-maps/extensor-carpi-ulnaris-muscle#1

Health Line. (2015c). Flexor carpi radialis. Recuperado el 17 de abril de 2019, de https://www.healthline.com/human-body-maps/flexor-carpi-radialis-muscle#1

Health Line. (2015d). Flexor carpi ulnaris. Recuperado el 17 de abril de 2019, de https://www.healthline.com/human-body-maps/flexor-carpi-ulnaris-muscle#1

Health Line. (2015e). Flexor digitorum profundus. Recuperado el 17 de abril de 2019, de https://www.healthline.com/human-body-maps/flexor-digitorum-profundus#1

Health Line. (2015f). Flexor digitorum superficialis. Recuperado el 17 de abril de 2019, de https://www.healthline.com/human-body-maps/flexor-digitorum-superficialis-muscle#1

Health Line. (2015g). Palmaris longus. Recuperado el 17 de abril de 2019, de https://www.healthline.com/human-body-maps/palmaris-longus-muscle#1

Health Line. (2015h). Pronator teres. Recuperado el 17 de abril de 2019, de https://www.healthline.com/human-body-maps/pronator-teres-muscle#1

Health Line. (2018). Arm Muscles Overview. Recuperado el 14 de abril de 2019, de https://www.healthline.com/human-body-maps/arm-muscles#upper-arm-muscles

Jadhav, S. D., & Channe, H. P. (2016). Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques. International Journal of Science and Research (IJSR), 5(1), 1842–1845. https://doi.org/10.21275/v5i1.nov153131

Karimpour, M., Parsaei, H., Sharifian, R., Rojhani, Z., & Yazdani, F. (2019). An Android Application for Estimating Muscle Onset Latency using Surface EMG Signal. Journal of Biomedical Physics and Engineering. https://doi.org/10.31661/jbpe.v0i0.700

Karlik, B. (2014). Machine Learning Algorithms for Characterization of EMG Signals. International Journal of Information and Electronics Engineering, 4(3). https://doi.org/10.7763/ijiee.2014.v4.433

Keil. (2019). STMicroelectronics STM32F103C8. Recuperado el 21 de abril de 2019, de http://www.keil.com/dd2/stmicroelectronics/stm32f103c8/#/eula-container

Levin, D. (2018). Humans beat robots, hands down. Recuperado el 1 de abril de 2019, de https://www.knowablemagazine.org/article/technology/2018/humans-beat-robots-hands-down?xid=PS_smithsonian

Liu, Y. W., Feng, F., & Gao, Y. F. (2014). HIT prosthetic hand based on tendon-driven mechanism. Journal of Central South University, 21(5), 1778–

1791. https://doi.org/10.1007/s11771-014-2124-z

Mahmoud Abduo, & Galster, M. (2015). Myo Gesture Control Armband for Medical Applications. Department of Computer Science and Software Engineering University of Canterbury, (October), 4–23. Recuperado de https://www.cosc.canterbury.ac.nz/research/reports/HonsReps/2015/hons_1502.pdf

Mannion, P. (2016). Myo armband : Wearables design focuses on packaging. EDN Network, 10.

Marin, G., Dominio, F., & Zanuttigh, P. (2014). HAND GESTURE RECOGNITION WITH LEAP MOTION AND KINECT DEVICES Department of Information Engineering , University of Padova. Image Processing (ICIP), 2014 IEEE International Conference on, 1565–1569. Recuperado de http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?arnumber=7025313&tag=1

McGuinness, H. (2018). Anatomy & Physiology (5a ed.). London: Hodder Education.

Medium. (2018). An Unofficial Introductory Tutorial to MyoWare Muscle Sensor Development Kit. Recuperado el 11 de abril de 2019, de https://medium.com/@leex5202/an-unofficial-introductory-tutorial-to-myoware-muscle-sensor-development-kit-e2169948e63

Mishra, V. K., Bajaj, V., Kumar, A., Sharma, D., & Singh, G. K. (2017). An efficient method for analysis of EMG signals using improved empirical mode decomposition. AEU - International Journal of Electronics and Communications, 72, 200–209. https://doi.org/10.1016/j.aeue.2016.12.008

Molchanov, P., Gupta, S., Kim, K., & Kautz, J. (2015). Hand gesture recognition with

3D convolutional neural networks. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2015–Octob, 1–7. https://doi.org/10.1109/CVPRW.2015.7301342

Montoya-Leal, V., Orozco-Duque, A., Ugarte, J. P., Portela, M., Franco, J. C., & Perez, V. Z. (2017). Assessment protocol of wrist flexion and extension to support processes in occupational health using Myo Armband, 60(October 2018). https://doi.org/10.1007/978-981-10-4086-3

Morales, L., & Pozo, D. (2017). An experimental comparative analysis among different classifiers applied to identify hand movements based on sEMG. 2017 IEEE 2nd Ecuador Technical Chapters Meeting, ETCM 2017, 2017–Janua, 1–6. https://doi.org/10.1109/ETCM.2017.8247504

MovilTronics. (2019). Bluetooth HC-06 Board. Recuperado el 21 de abril de 2019, de https://moviltronics.com.co/varios/74-bluetooth-hc-06-board.html

MyoBridge Firmware. (2019). System Status Constants. Recuperado el 30 de abril de 2019, de https://htmlpreview.github.io/?https://raw.githubusercontent.com/vroland/MyoBridge/master/myobridge_firmware/Documentation/html/group__myb__system__status.html#gaf2d9e63dcf3d3f292d2a167721ef0d34

MyoBridge Library. (2019a). Functions for Basic Usage. Recuperado el 2 de mayo de 2019, de https://htmlpreview.github.io/?https://raw.githubusercontent.com/vroland/MyoBridge/master/Arduino/libraries/MyoBridge/extras/Documentation/html/group__basic__funcs.html#ga6d564f122e4fd5e3f3d968826ad21359

MyoBridge Library. (2019b). MyoBridge Class Reference. Recuperado el 2 de mayo de 2019, de https://htmlpreview.github.io/?https://raw.githubusercontent.com/vroland

/MyoBridge/master/Arduino/libraries/MyoBridge/extras/Documentation/html/class_myo_bridge.html#gacba3fc3a629cfe244feff21f521bff00

Naylamp Mechatronics. (2019). Convertidor Voltaje DC-DC Step-Down 2A MP1584EN. Recuperado el 21 de abril de 2019, de https://naylampmechatronics.com/conversores-dc-dc/85-convertidor-voltaje-dc-dc-step-down-2a-mp1584en.html

Pololu. (2019). MyoWare Muscle Sensor. Recuperado el 11 de abril de 2019, de https://www.pololu.com/product/2732

Prometec. (2019). ARDUINO Y LOS TIMERS. Recuperado el 1 de mayo de 2019, de https://www.prometec.net/timers/#

Raj, A. (2018). Getting Started with STM32 using Arduino IDE: Blinking LED. Recuperado el 2 de mayo de 2019, de https://circuitdigest.com/microcontroller-projects/getting-started-with-stm32-development-board-stm32f103c8-using-arduino-ide

Rawat, S., Vats, S., & Kumar, P. (2016). Evaluating and exploring the MYO ARMBAND. Proceedings of the 5th International Conference on System Modeling and Advancement in Research Trends, SMART 2016, 115–120. https://doi.org/10.1109/SYSMART.2016.7894501

RedBearLab. (2017). CCLoader. Recuperado el 26 de abril de 2019, de https://github.com/RedBearLab/CCLoader

RIBAS Medicina. (2019). Electrodo Desechable Foam Gel Sólido GS3648 (Bolsa 50u). Recuperado el 3 de abril de 2019, de https://www.ribasmedicina.com/producto/electrodo-desechable-foam-gel-solido-gs3648-bolsa-50u/

Richer, R., Blank, P., Schuldhaus, D., & Eskofier, B. M. (2014). Real-time ECG and EMG analysis for biking using android-based mobile devices. Proceedings - 11th International Conference on Wearable and Implantable Body Sensor Networks, BSN 2014, 104–108. https://doi.org/10.1109/BSN.2014.20

ROBOTIQ. (2019). 3-Finger Adaptive Robot Gripper. Recuperado el 1 de abril de 2019, de https://robotiq.com/products/3-finger-adaptive-robot-gripper

Roland, V. (2017). MyoBridge. Recuperado el 26 de abril de 2019, de https://github.com/vroland/MyoBridge

ROS Components. (2019). Shadow dexterous. Recuperado el 2 de abril de 2019, de https://www.roscomponents.com/en/robotic-hands/117-shadow-dexterous-robotic-hand.html?search_query=hand&results=21

Ryan, M., Metz, C., & Taylor, R. (2018). How Robot Hands Are Evolving to Do What Ours Can. Recuperado el 1 de abril de 2019, de https://www.nytimes.com/interactive/2018/07/30/technology/robot-hands.html

Sathiyanarayanan, M., Mulling, T., & Nazir, B. (2015). Controlling a Robot Using a Wearable Device (MYO). International Journal of Engineering Development and Research (www.ijedr.org), 3(1), 2321–9939.

Sathiyanarayanan, M., & Rajan, S. (2016). MYO Armband for physiotherapy healthcare: A case study using gesture recognition application. 2016 8th International Conference on Communication Systems and Networks, COMSNETS 2016, 1–6. https://doi.org/10.1109/COMSNETS.2016.7439933

SCHUNK. (2019). SVH Hand. Recuperado el 1 de abril de 2019, de

https://schunk.com/ru_ru/zakhvatnye-sistemy/jarkie-momenty/svh/

Seeed Studio. (2019). Grove - EMG Detector. Recuperado el 12 de abril de 2019, de https://www.seeedstudio.com/Grove-EMG-Detector-p-1737.html

Shin, H.-S., Ganiev, A., & Lee, K.-H. (2015). Design of a Virtual Robotic Arm based on the EMG variation. Advanced Science and Technology Letters, 113, 38–43. https://doi.org/10.14257/astl.2015.113.09

Shroffe, E. H., & Manimegalai, P. (2013). HAND GESTURE RECOGNITION BASED ON EMG SIGNALS USING ANN. IEEE International Conference on Consumer Electronics - Berlin, ICCE-Berlin, 2013–April(3), 174–178. https://doi.org/10.1109/ICCE-Berlin.2016.7684748

Stern, B. (2018). Myo Armband Teardown. Adafruit learning system, 7.

Stm32duino. (2019). Blue Pill. Recuperado el 21 de abril de 2019, de https://wiki.stm32duino.com/index.php?title=Blue_Pill

STMicroelectronics. (2018). RM0008 Reference manual.

STMicroelectronics. (2019). STM32CubeMX. Recuperado el 2 de mayo de 2019, de https://www.st.com/en/development-tools/stm32cubemx.html

Support GetMyo. (2019). What are the different LED status descriptions for the Myo armband? Recuperado el 16 de abril de 2019, de https://support.getmyo.com/hc/en-us/articles/202724369-What-are-the-different-LED-status-descriptions-for-the-Myo-armband-

Tatarian, K., Parente, E., Couceiro, M., & Faria, D. (2018). Stepping-stones to Transhumanism: An EMG- controlled Low-cost Prosthetic Hand for Academia. International Conference on Intelligent Systems, (September).

Tsipouras, M. G. (2018). Uterine EMG Signals Spectral Analysis for Pre-Term Birth Prediction, 8(5), 3310–3315.

Visconti, P., Gaetani, F., Zappatore, G. A., & Primiceri, P. (2018). Technical features and functionalities of Myo armband: An overview on related literature and advanced applications of myoelectric armbands mainly focused on arm prostheses. International Journal on Smart Sensing and Intelligent Systems, 11(1), 1–25. https://doi.org/10.21307/ijssis-2018-005

Youbionic. (2019). Youbionic Hand 2019. Recuperado el 11 de abril de 2019, de https://www.youbionic.com/store/youbionic-hand-2019

Xataca. (2015). Las tres leys de Clarke. Recuperado el 11 de abril de 2019, de https://www.xatakaciencia.com/sabias-que/las-tres-leyes-de-clarke

ANEXOS

Anexo 1: Manual de usuario del sistema de control de la mano robótica.

El presente manual de usuario tiene como objetivo indicar la forma adecuada de utilizar el sistema microcontrolado y también explicar los diferentes eventos relacionados al mismo.

1. Posición del sensor MYO Armband.

El sensor MYO Armband debe estar colocado en la tercera parte del antebrazo derecho, en el área cercana a la articulación del codo con el logotipo de Thalmic Labs apuntando en línea recta al dedo medio de la mano, como se muestra en la Figura 1.

Figura 1. Colocación del MYO Armband en el usuario.

2. Inicialización del MYO Armband.

El MYO Armband debe estar adaptado al calor corporal del usuario para poder capturar señales electromiográficas correctas. Para ello se debe conectar y desconectar el sensor al computador por medio del cable USB para encenderlo.

Posteriormente, se inserta el adaptador bluetooth propio del MYO Armband en la PC y se abre la aplicación “Armband Manager”. Dentro de esta, se selecciona el MYO Armband y se presiona el botón “Connect” para enlazar el sensor al computador, como se muestra en la Figura 2.

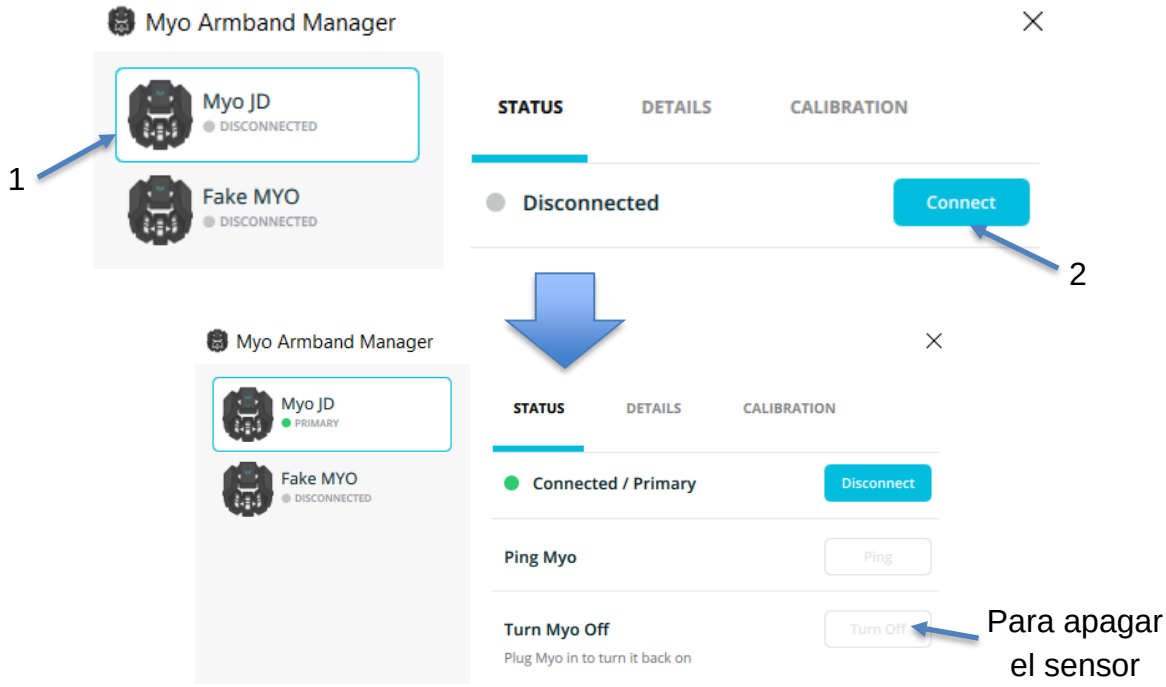


Figura 2. Conexión del sensor con la aplicación Armband Manager.

Se debe mover la palma de la mano hacia afuera para sincronizarlo al programa. Después, en el computador aparecerá un mensaje indicando que el sensor se está calentando. Se espera a que este mensaje de estado cambie, como se observa en la Figura 3.

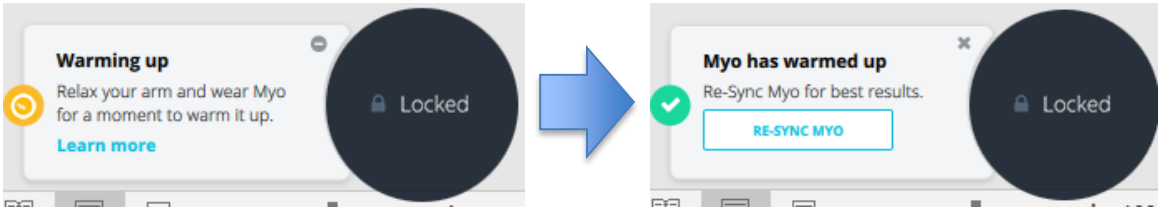


Figura 3. Calentamiento del MYO Armband en el usuario.

Finalmente, se debe desenlazar el sensor al computador presionando el botón “Disconnect” de la interfaz de la aplicación y retirando el adaptador bluetooth.

Al momento de hacer esto, el logotipo de Thalmic Labs comenzará a parpadear lentamente mientras que el LED indicador se encontrará apagado. Esto significa que el sensor ya no se encuentra vinculado a su módulo bluetooth, pero que ya está inicializado.

Es importante mencionar que se puede omitir este paso simplemente encendiendo el MYO Armband y esperando alrededor de 2 minutos (tiempo promedio de calentamiento), sin conectarlo al computador.

Sin embargo, es recomendable realizar esto al menos en la primera vez que se utilice el sensor para tener la seguridad de que se encuentra apto para funcionar correctamente.

3. Enlace con el sistema de control.

Se debe conectar el sistema microcontrolado a una fuente de entrada de 8.4V. Después, mediante el switch que se encuentra en la parte posterior de la mano robótica, se enciende el sistema y se espera (alrededor de 15 segundos) a que este se enlace con el MYO Armband, como se indica en la Figura 4.

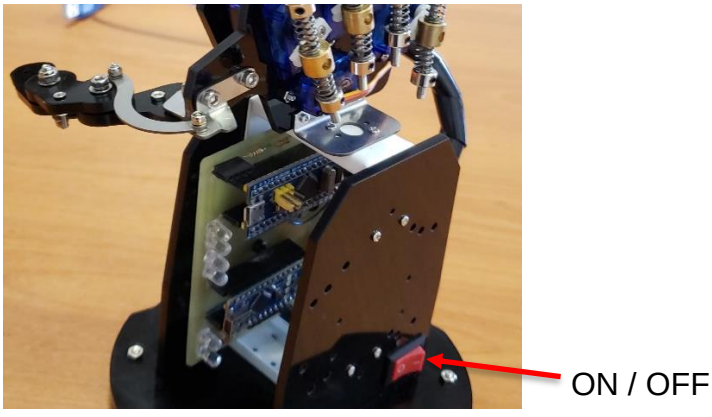


Figura 4. Encendido del sistema de control.

Cuando esto sucede, el LED indicador del sensor se mantendrá en azul mientras que el logotipo de Thalmic Labs comenzará a parpadear lentamente, así como también el LED verde del sistema se encenderá por un segundo indicando que el sistema está listo para funcionar, como se muestra en la Figura 5.

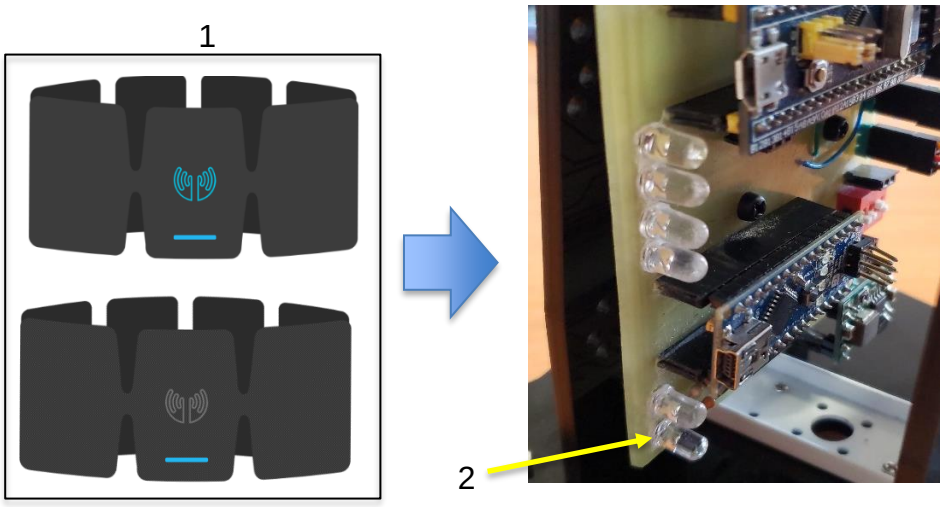


Figura 5. Enlace del sistema con el sensor MYO Armband.

IMPORTANTE: Si al momento de realizar este paso, el sensor no se conecta con el sistema y comienza a mantener el logotipo de Thalmic Labs encendido mientras que el LED indicador se conserva apagado, puede deberse a que existe alguna interferencia en el medio que impide la comunicación entre ambas partes. También puede deberse a que se realizó accidentalmente el movimiento de sincronización antes de que se realice la conexión al sistema.

Para mitigar este inconveniente se debe restablecer el MYO Armband, conectándolo por un segundo a la PC. Así como también apagando el sistema por dos segundos y después volviéndolo a encender. Después, se debe acercar el sensor a la placa para que pueda ser reconocido y conectado.

Este proceso debe realizarse las veces que sean necesarias hasta que el sensor se enlace como se muestra en la Figura 5, caso contrario no se podrá continuar.

4. Utilización del sistema de control de la mano robótica.

Una vez que el sensor se conecte y el sistema se encuentre listo para operar, se debe presionar el botón RESET del STM32, como se muestra en la Figura 6. Esto hará que el sistema comience a recolectar las señales electromiográficas y a predecir los movimientos de mano abierta y mano cerrada.

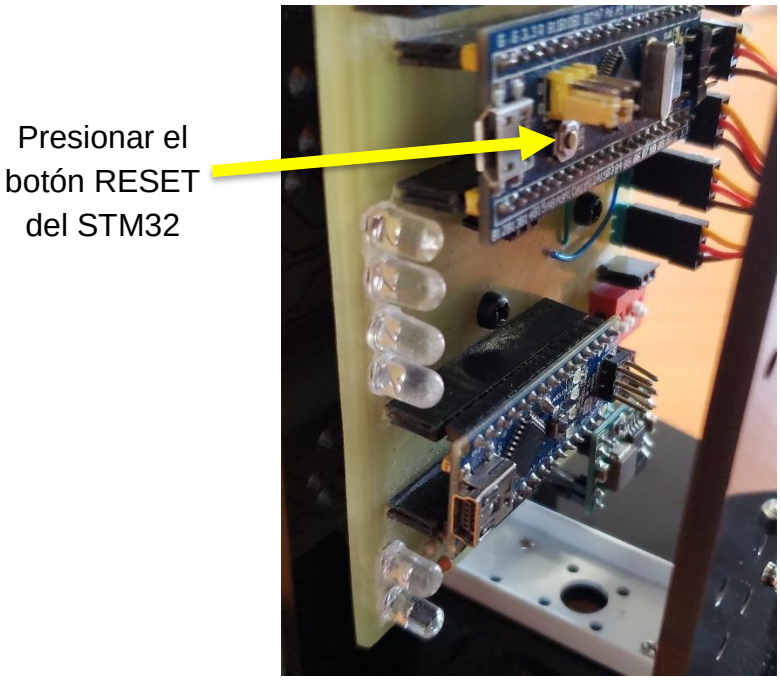
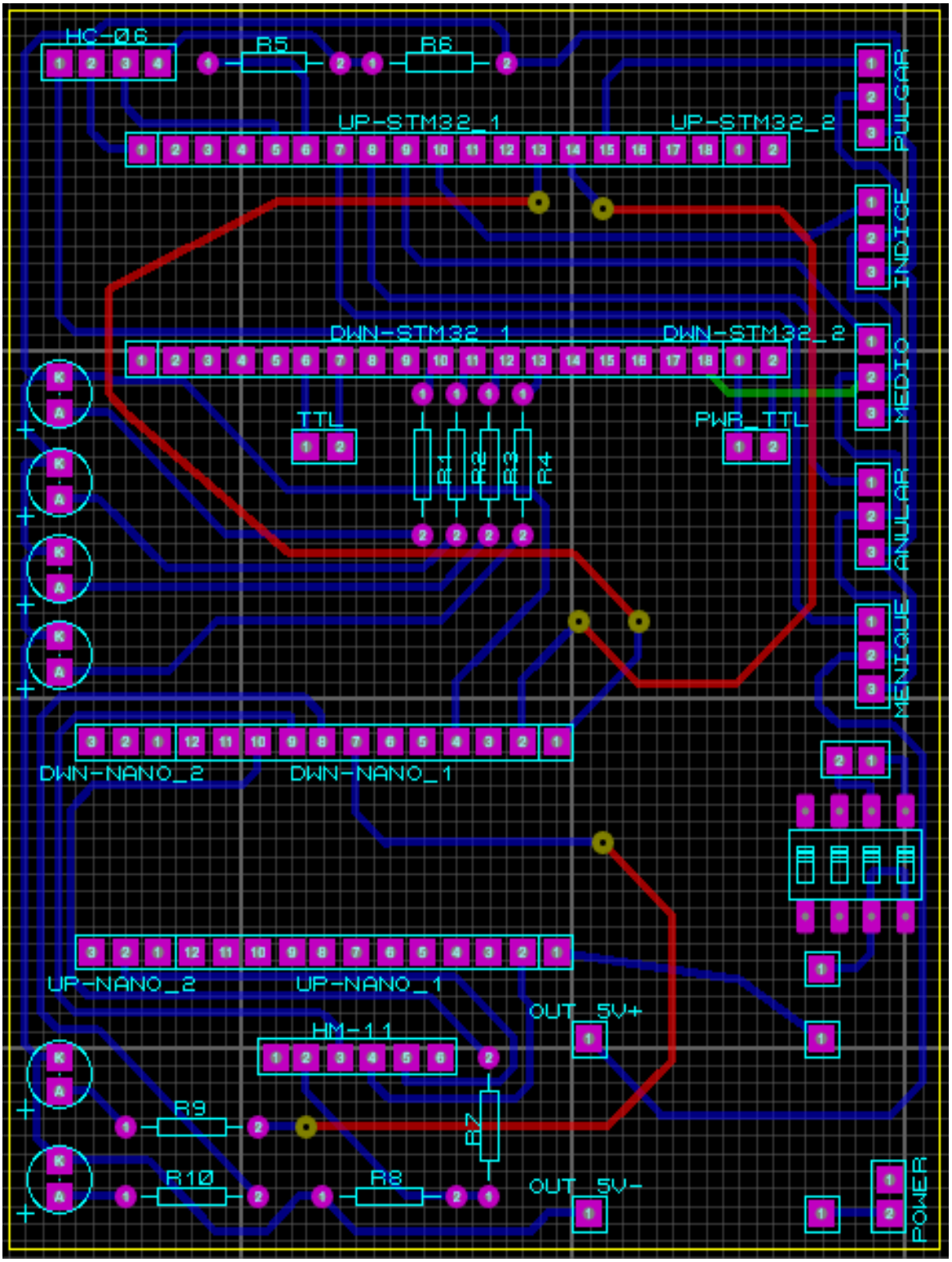


Figura 6. Iniciar el trabajo del sistema de control.

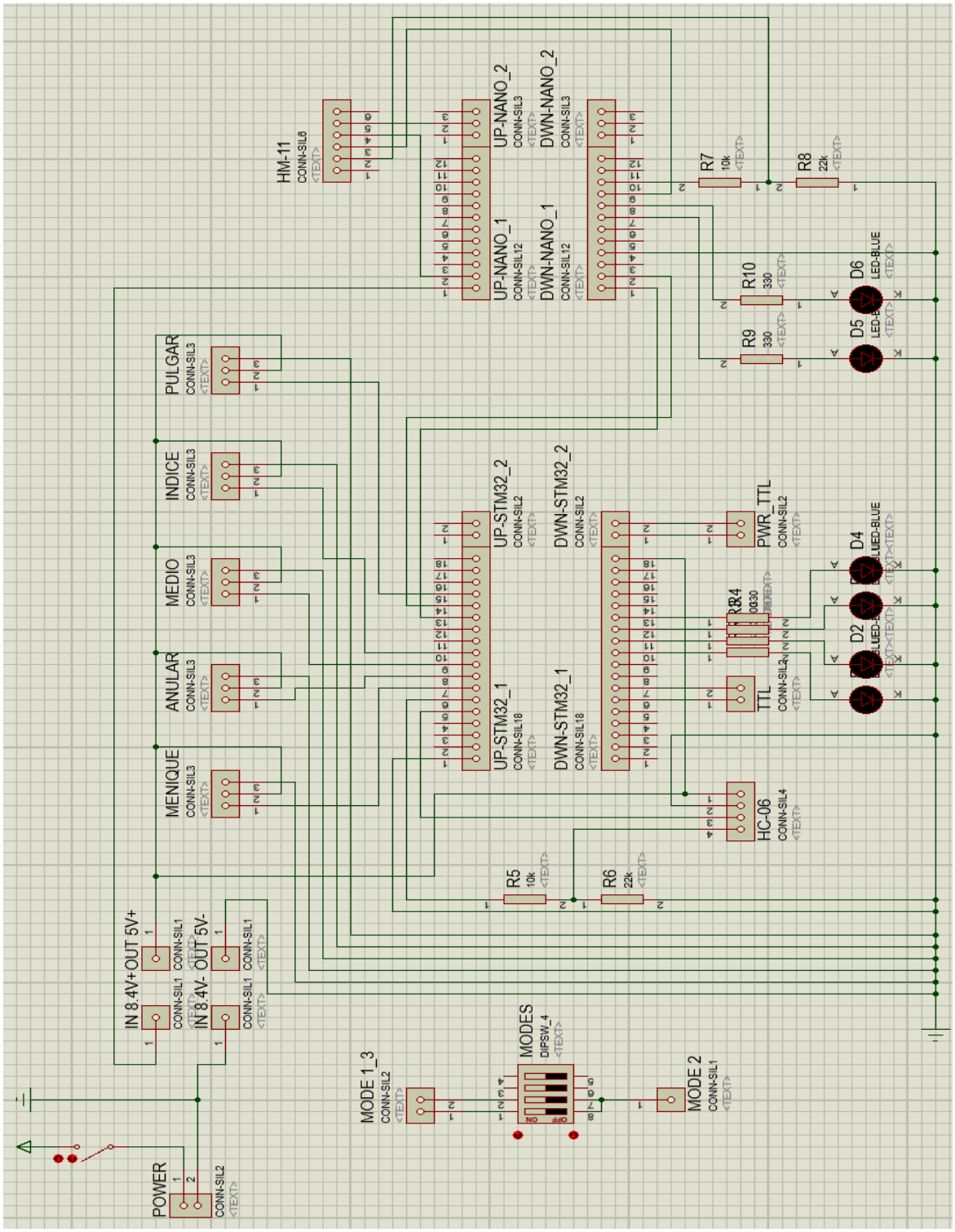
Finalmente, el sistema comenzará automáticamente a predecir los movimientos que el usuario realice con la mano. Depende el gesto detectado, se abrirá o se cerrará la mano robótica. En caso de que el movimiento no sea identificado, el sistema mantendrá la última acción reconocida.

Al terminar la utilización del sistema de control, se debe apagarlo mediante el switch mostrado en la Figura 4. De forma automática se desconectará el sensor, emitiendo una serie de vibraciones y mediante la aplicación Armband Manager se apaga el sensor presionando el botón “Turn Off” como se expresa en la Figura 2.

Anexo 2: Placa del sistema microcontrolado.



Anexo 3: Circuito del sistema microcontrolado.



Anexo 4: Tablas de las pruebas realizadas a usuarios entrenados para mano abierta.

Tabla 1.

Resultados de la predicción de mano abierta con ventanas EMG de 30x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Abierto	Cerrado	NI
Usuario 1	711.83	30.78	18.57	15	6	4
Usuario 2	743.93	30.72	19.37	23	2	0
Usuario 3	724.24	30.56	18.87	25	0	0
Usuario 4	719.90	30.68	18.76	23	2	0
Usuario 5	760.15	30.59	19.77	25	0	0
Usuario 6	776.18	30.55	20.17	25	0	0
Usuario 7	759.94	30.63	19.76	22	3	0

Tabla 2.

Resultados de la predicción de mano abierta con ventanas EMG de 50x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Abierto	Cerrado	NI
Usuario 1	1166.98	35.45	30.06	25	0	0
Usuario 2	1211.04	35.32	31.16	25	0	0
Usuario 3	1216.00	35.08	31.28	24	0	1
Usuario 4	1165.34	35.21	30.01	17	7	1
Usuario 5	1295.10	35.19	33.26	21	4	0
Usuario 6	1201.12	35.38	33.41	23	1	1
Usuario 7	1239.01	35.15	31.85	25	0	0

Tabla 3.

Resultados de la predicción de mano abierta con ventanas EMG de 70x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Abierto	Cerrado	NI
Usuario 1	1758.07	40.24	44.96	19	6	0
Usuario 2	1677.67	40.05	42.94	25	0	0
Usuario 3	1758.17	39.70	44.95	14	11	0
Usuario 4	1782.16	40.12	45.56	19	2	4
Usuario 5	1816.36	39.94	44.41	25	0	0
Usuario 6	1762.68	40.22	42.07	20	4	1
Usuario 7	1634.07	40.01	41.85	24	1	0

Tabla 4.

Resultados de la predicción de mano abierta con ventanas EMG de 100x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Abierto	Cerrado	NI
Usuario 1	2365.90	46.95	60.32	17	7	1
Usuario 2	2567.23	46.93	65.35	23	1	1
Usuario 3	2357.42	46.35	60.09	6	15	4
Usuario 4	2477.07	46.83	63.10	20	5	0
Usuario 5	2263.65	46.78	57.76	21	4	0
Usuario 6	2541.37	47.10	60.55	12	5	8
Usuario 7	2415.14	46.57	61.54	20	2	3

Anexo 5: Tablas de las pruebas realizadas a usuarios entrenados para mano cerrada.

Tabla 1.

Resultados de la predicción de mano cerrada con ventanas EMG de 30x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Cerrado	Abierto	NI
Usuario 1	711.80	30.87	18.57	21	1	3
Usuario 2	735.91	30.49	19.21	24	0	1
Usuario 3	723.97	30.62	18.86	25	0	0
Usuario 4	659.81	30.84	17.27	14	11	0
Usuario 5	791.88	30.71	20.56	24	0	1
Usuario 6	767.87	30.79	19.97	22	3	0
Usuario 7	777.48	30.70	20.20	14	8	3

Tabla 2.

Resultados de la predicción de mano cerrada con ventanas EMG de 50x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Cerrado	Abierto	NI
Usuario 1	1274.77	35.55	32.76	20	5	0
Usuario 2	1178.78	35.60	30.36	25	0	0
Usuario 3	1195.83	35.25	30.78	25	0	0
Usuario 4	1131.12	35.33	29.16	10	9	6
Usuario 5	1258.84	35.57	32.36	24	1	0
Usuario 6	1370.71	35.54	35.16	20	5	0
Usuario 7	1246.86	35.46	32.06	13	7	5

Tabla 3.

Resultados de la predicción de mano cerrada con ventanas EMG de 70x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Cerrado	Abierto	NI
Usuario 1	1684.21	40.56	43.12	25	0	0

Usuario 2	1673.73	40.36	42.85	23	1	1
Usuario 3	1830.08	39.89	46.75	25	0	0
Usuario 4	1649.82	40.33	42.25	24	1	0
Usuario 5	1853.58	40.39	47.35	24	1	0
Usuario 6	2093.42	40.52	53.35	23	2	0
Usuario 7	1625.62	40.40	41.65	16	9	0

Tabla 4.

Resultados de la predicción de mano cerrada con ventanas EMG de 100x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Cerrado	Abierto	NI
Usuario 1	2438.62	47.23	62.15	24	1	0
Usuario 2	2566.33	47.38	65.34	14	11	0
Usuario 3	2403.00	46.87	61.25	21	2	2
Usuario 4	2330.57	47.18	59.44	15	10	0
Usuario 5	2370.61	47.25	60.45	22	3	0
Usuario 6	2346.81	47.21	60.31	12	4	9
Usuario 7	2366.62	47.23	60.35	10	1	14

Anexo 6: Tablas de las pruebas realizadas a usuarios entrenados para movimiento no identificado.

Tabla 1.

Resultados de las veces que el sistema no identificó movimiento alguno. Prueba realizada con ventanas EMG de 30x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	NI	Abierto	Cerrado

Usuario 1	688.57	30.33	17.97	16	0	9
Usuario 2	720.67	30.26	18.77	19	1	5
Usuario 3	788.30	30.30	20.46	15	2	8
Usuario 4	702.61	30.21	18.32	18	7	0
Usuario 5	712.18	30.25	18.56	5	12	8
Usuario 6	724.40	30.51	18.87	10	5	10
Usuario 7	708.72	30.44	18.48	7	15	3

Tabla 2.

Resultados de las veces que el sistema no identificó movimiento alguno. Prueba realizada con ventanas EMG de 50x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	NI	Abierto	Cerrado
Usuario 1	1124.17	35.57	30.97	22	1	2
Usuario 2	1251.73	35.61	32.16	6	19	0
Usuario 3	1335.60	35.78	34.26	8	5	12
Usuario 4	1149.42	34.57	29.6	16	3	6
Usuario 5	1316.16	35.57	33.77	18	4	2
Usuario 6	1157.39	34.80	29.80	18	2	5
Usuario 7	1135.84	35.78	29.27	20	0	5

Tabla 3.

Resultados de las veces que el sistema no identificó movimiento alguno. Prueba realizada con ventanas EMG de 70x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	NI	Abierto	Cerrado
Usuario 1	1559.10	39.36	39.96	25	0	0
Usuario 2	1747.27	39.11	44.66	24	0	1
Usuario 3	1858.78	39.37	47.45	20	5	0

Usuario 4	1605.67	39.30	41.12	19	2	4
Usuario 5	2059.17	39.07	52.46	25	0	0
Usuario 6	1638.99	39.42	41.96	25	0	0
Usuario 7	1655.14	39.41	42.36	24	1	0

Tabla 4.

Resultados de las veces que el sistema no identificó movimiento alguno. Prueba realizada con ventanas EMG de 100x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	NI	Abierto	Cerrado
Usuario 1	2288.13	45.95	58.35	25	0	0
Usuario 2	2500.51	45.49	63.65	24	1	0
Usuario 3	2327.85	45.82	59.34	18	0	7
Usuario 4	2305.72	45.96	58.79	20	4	1
Usuario 5	2208.32	45.37	56.34	25	0	0
Usuario 6	2319.33	45.91	59.13	24	0	1
Usuario 7	2407.12	46.15	61.33	21	0	4

Anexo 7: Tablas de las pruebas realizadas a usuarios no entrenados para mano abierta.

Tabla 1.

Resultados de la predicción de mano abierta con ventanas EMG de 30x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Abierto	Cerrado	NI
Usuario 1	735.72	30.87	19.16	20	2	3
Usuario 2	716.01	30.64	18.67	25	0	0

Usuario 3	715.81	30.77	18.66	25	0	0
Usuario 4	764.74	30.61	19.88	22	2	1

Tabla 2.

Resultados de la predicción de mano abierta con ventanas EMG de 50x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Abierto	Cerrado	NI
Usuario 1	1166.52	35.64	30.05	19	3	3
Usuario 2	1203.23	35.32	30.96	21	0	4
Usuario 3	1171.22	35.21	30.16	23	1	1
Usuario 4	1339.47	35.16	34.37	25	0	0

Tabla 3.

Resultados de la predicción de mano abierta con ventanas EMG de 70x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Abierto	Cerrado	NI
Usuario 1	1685.72	40.36	43.15	25	0	0
Usuario 2	1697.50	39.87	43.43	20	4	1
Usuario 3	1721.64	40.30	44.05	24	0	1
Usuario 4	1857.78	40.16	47.45	21	3	1

Tabla 4.

Resultados de la predicción de mano abierta con ventanas EMG de 100x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Abierto	Cerrado	NI
Usuario 1	2512.06	46.84	63.97	24	0	1
Usuario 2	2423.89	46.65	61.76	24	1	0
Usuario 3	2330.91	47.00	59.45	6	9	10
Usuario 4	2508.20	46.88	63.88	10	1	14

Anexo 8: Tablas de las pruebas realizadas a usuarios no entrenados para mano cerrada.

Tabla 1.

Resultados de la predicción de mano cerrada con ventanas EMG de 30x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Cerrado	Abierto	NI
Usuario 1	703.87	30.87	18.37	25	0	0
Usuario 2	719.98	30.63	18.77	23	1	1
Usuario 3	695.72	30.89	18.17	19	3	3
Usuario 4	803.74	30.90	20.87	21	3	1

Tabla 2.

Resultados de la predicción de mano cerrada con ventanas EMG de 50x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Cerrado	Abierto	NI
Usuario 1	1170.75	35.67	30.16	24	1	0
Usuario 2	1186.95	35.44	30.56	20	4	1
Usuario 3	1186.85	35.66	30.56	20	4	1
Usuario 4	1319.86	35.59	33.89	22	2	1

Tabla 3.

Resultados de la predicción de mano cerrada con ventanas EMG de 70x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Cerrado	Abierto	NI
Usuario 1	1729.67	40.51	44.25	22	3	0
Usuario 2	1650.58	40.34	42.27	13	7	5
Usuario 3	1657.66	40.55	42.46	17	6	2

Usuario 4	1809.61	40.55	46.25	21	2	2

Tabla 4.

Resultados de la predicción de mano cerrada con ventanas EMG de 100x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	Cerrado	Abierto	NI
Usuario 1	2466.15	47.63	62.84	14	8	3
Usuario 2	2438.83	47.07	62.15	17	4	4
Usuario 3	2326.44	47.39	59.35	17	5	3
Usuario 4	2566.64	47.23	65.35	14	2	9

Anexo 9: Tablas de las pruebas realizadas a usuarios no entrenados para movimiento no identificado.

Tabla 1.

Resultados de las veces que el sistema no identificó movimiento alguno. Prueba realizada con ventanas EMG de 30x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	NI	Abierto	Cerrado
Usuario 1	765.44	30.42	19.90	20	4	1
Usuario 2	699.81	30.23	18.25	10	7	8
Usuario 3	696.50	30.25	18.17	5	0	20
Usuario 4	764.32	30.48	19.87	15	7	3

Tabla 2.

Resultados de las veces que el sistema no identificó movimiento alguno. Prueba realizada con ventanas EMG de 50x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	NI	Abierto	Cerrado
Usuario 1	1314.91	35.71	33.74	5	14	6
Usuario 2	1132.23	35.55	29.17	24	1	0
Usuario 3	1151.98	35.65	29.67	17	6	2
Usuario 4	1232.13	35.62	31.67	24	0	1

Tabla 3.

Resultados de las veces que el sistema no identificó movimiento alguno. Prueba realizada con ventanas EMG de 70x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	NI	Abierto	Cerrado
Usuario 1	1930.85	39.38	46.26	15	9	1
Usuario 2	1575.32	39.20	40.36	25	0	0
Usuario 3	1611.29	39.17	41.26	24	1	0
Usuario 4	1802.84	39.44	46.06	22	2	1

Tabla 4.

Resultados de las veces que el sistema no identificó movimiento alguno. Prueba realizada con ventanas EMG de 100x8.

GRUPO	TPA (ms)	TPR (ms)	TT (s)	NI	Abierto	Cerrado
Usuario 1	2315.67	45.81	59.04	19	4	2
Usuario 2	2304.51	45.65	58.75	25	0	0
Usuario 3	2316.42	45.68	59.05	25	0	0
Usuario 4	2371.93	45.83	60.44	13	12	0

Anexo 10: Matrices de confusión para las ventanas EMG de 30x8.

Tabla 1.

Matriz de confusión de las pruebas realizadas a los usuarios entrenados.

Matriz de confusión: 30x8		**Predicción**		
		Abierto	**Cerrado**	**No identificado**
Real	**Abierto**	158	13	4
	Cerrado	23	144	8
	No identificado	42	43	90

Tabla 2.

Matriz de confusión de las pruebas realizadas a los usuarios no entrenados.

Matriz de confusión: 30x8		**Predicción**		
		Abierto	**Cerrado**	**No identificado**
Real	**Abierto**	92	4	4
	Cerrado	7	88	5
	No identificado	18	32	50

Anexo 11: Matrices de confusión para las ventanas EMG de 50x8.

Tabla 1.

Matriz de confusión de las pruebas realizadas a los usuarios entrenados.

Matriz de confusión: 50x8		Predicción		
		Abierto	**Cerrado**	**No identificado**
Real	**Abierto**	160	12	3
	Cerrado	27	137	11
	No identificado	34	32	108

Tabla 2.

Matriz de confusión de las pruebas realizadas a los usuarios no entrenados.

Matriz de confusión: 50x8		Predicción		
		Abierto	**Cerrado**	**No identificado**
Real	**Abierto**	88	4	8
	Cerrado	11	86	3
	No identificado	21	9	70

Anexo 12: Matrices de confusión para las ventanas EMG de 70x8.

Tabla 1.

Matriz de confusión de las pruebas realizadas a los usuarios entrenados.

Matriz de confusión: 70x8		Predicción		
		Abierto	**Cerrado**	**No identificado**
Real	**Abierto**	146	24	5
	Cerrado	14	160	0
	No identificado	8	5	162

Tabla 2.

Matriz de confusión de las pruebas realizadas a los usuarios no entrenados.

Matriz de confusión: 70x8		Predicción		
		Abierto	**Cerrado**	**No identificado**
Real	**Abierto**	90	7	3
	Cerrado	18	73	9
	No identificado	12	2	86

Anexo 13: Matrices de confusión para las ventanas EMG de 100x8.

Tabla 1.

Matriz de confusión de las pruebas realizadas a los usuarios entrenados.

Matriz de confusión: 100x8		Predicción		
		Abierto	**Cerrado**	**No identificado**
Real	**Abierto**	119	39	17
	Cerrado	32	118	25
	No identificado	5	13	157

Tabla 2.

Matriz de confusión de las pruebas realizadas a los usuarios no entrenados.

Matriz de confusión: 100x8		Predicción		
		Abierto	**Cerrado**	**No identificado**
Real	**Abierto**	64	11	25
	Cerrado	19	62	19
	No identificado	16	2	82