# UNIVERSIDAD UTE

## FACULTAD DE CIENCIAS DE LA INGENIERÍA E INDUSTRIAS

## CARRERA DE INGENIERÍA MECATRÓNICA

# DISEÑO DE UN CONTROLADOR NEURONAL DE FUERZA PARA UN EXOESQUELETO DE EXTREMIDAD INFERIOR

**TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICO**

**CÉSAR FRANCISCO MOYA GUERRA**

**DIRECTOR: PhD. BONILLA VENEGAS FÉLIX VLADIMIR**

**Quito, septiembre 2021**

# FORMULARIO DE REGISTRO BIBLIOGRÁFICO

# PROYECTO DE TITULACIÓN

| DATOS DE CONTACTO | |
|---|---|
| **CÉDULA DE IDENTIDAD**: | 1719000125 |
| **APELLIDO Y NOMBRES:** | Moya Guerra César Francisco |
| **DIRECCIÓN:** | Av. Diego Vaca de la Vega y Santa Teresa |
| **EMAIL:** | cesar_fmg@hotmail.com |
| **TELÉFONO FIJO:** | 02-2494922 |
| **TELÉFONO MOVIL:** | 0983171827 |

| DATOS DE LA OBRA | |
|---|---|
| **TÍTULO**: | Diseño de un controlador neuronal de fuerza para un exoesqueleto de extremidad inferior |
| **AUTOR O AUTORES**: | Moya Guerra César Francisco |
| **FECHA DE ENTREGA DEL PROYECTO DE TITULACIÓN**: | 02/12/2021 |
| **DIRECTOR DEL PROYECTO DE TITULACIÓN:** | Bonilla Venegas Félix Vladimir, PhD. |
| **PROGRAMA** | **PREGRADO** X **POSGRADO** |
| **TÍTULO POR EL QUE OPTA:** | Ingeniero en Mecatrónica |
| **RESUMEN: Mínimo 250 palabras** | El presente proyecto propone el diseño de un controlador neuronal de fuerza para un exoesqueleto de extremidad inferior. En este proyecto se aplicó el modelo en V para el diseño del sistema de control para sistemas Mecatrónicos. En la etapa inicial de este proyecto se definió los requerimientos del sistema de control, los cuales delimitan los parámetros de la estructura de este. En estos requerimientos se marcan el lenguaje de programación en C/C++ y las placas de desarrollo a usarse en el avance del proyecto. Posteriormente, se realizó la definición de las funciones específicas de un control con red neuronal artificial previamente entrenada, donde se determinó el funcionamiento de cada etapa del entorno de simulación el cual se encontraba en el software Matlab-Simulink, así como el tipo de dato de entrada el cual son señales electromiográficas que son generadas por los músculos del ser humano. Con esto se implementó un diseño de alto nivel, |

| | |
|---|---|
| | definiendo e implementando cada bloque de control del entorno de Simulink necesario para el sistema, el cual se encarga de procesar las señales que entra al sistema. Todo se realiza para la implementación del sistema de control en las diferentes placas de control, esto a razón de poseer distintos tipos de estructura en su arquitectura y en específico de tener implementado diferentes microcontroladores. Esto por motivo de observar y analizar el comportamiento de cada microcontrolador implementado en la placa de control, y dar a conocer el mejor comportamiento para entender el escenario ideal a usar cada una de estas placas. Todo esto se realizó con el método PIL, el cual permite generar código programable para distintas placas de desarrollo. Para la etapa final, se realiza una observación y análisis profundo de cada comportamiento, tomando registro posterior al procesamiento de diferentes señales electromiográficas. |
| **PALABRAS CLAVES:** | Placa de desarrollo, PIL, red neuronal artificial, C/C++, señales electromiográficas, procesador, entorno de control, software Matlab-Simulink, tiempo. |
| **ABSTRACT:** | The present project proposes the design of a neural force controller for a lower extremity exoskeleton. In this project, the V-model was applied for the design of the control system for Mechatronic systems. In the initial stage of this project, the requirements of the control system were defined, which delimit the parameters of its structure. These requirements mark the programming language in C / C + + and the development boards to be used in the progress of the project. Subsequently, the definition of the specific functions of a control with previously trained artificial neural network was carried out, where the operation of each stage of the simulation environment which was in the Matlab-Simulink software was determined, as well as the type of input data which are electromyographic signals that are generated by the muscles of the human being. With this, a high-level design was implemented, |

| | |
|---|---|
| | defining, and implementing each control block of the Simulink environment necessary for the system, which is responsible for processing the signals that enter the system. Everything is done for the implementation of the control system in the different control boards, this because of having different types of structure in its architecture and specifically having implemented different microcontrollers. This is for the purpose of observing and analyzing the behavior of each microcontroller implemented in the control board, and to publicize the best behavior to understand the ideal scenario to use each of these boards. All this was done with the PIL method, which allows to generate programmable code for different development boards. For the final stage, an in-depth observation and analysis of each behavior is carried out, taking record after the processing of different electromyographic signals. |
| **KEYWORDS** | Development board, PIL, artificial neural network, C / C++, electromyographic signals, processor, control environment, Matlab-Simulink software, time. |

Se autoriza la publicación de este Proyecto de Titulación en el Repositorio Digital de la Institución.

f:_____

MOYA GUERRA CÉSAR FRANCISCO

1719000125

# DECLARACIÓN Y AUTORIZACIÓN

Yo, MOYA GUERRA CÉSAR FRANCISCO, CI 1719000125 autor del proyecto titulado: Diseño de un controlador neuronal de fuerza para un exoesqueleto de extremidad inferior previo a la obtención del título de INGENIERÍA EN MECATRÓNICA en la Universidad UTE.

1. Declaro tener pleno conocimiento de la obligación que tienen las Instituciones de Educación Superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Autorizo a la BIBLIOTECA de la Universidad UTE a tener una copia del referido trabajo de graduación con el propósito de generar un Repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Quito, 21 de septiembre de 2021.

f:_____

MOYA GUERRA CÉSAR FRANCISCO
1719000125

# CERTIFICACIÓN DEL TUTOR

En mi calidad de tutor de tesis de grado, certifico que el presente trabajo que lleva por título Diseño de un controlador neuronal de fuerza para un exoesqueleto de extremidad inferior para aspirar al título de INGENIERÍA EN MECATRÓNICA fue desarrollado por Moya Guerra César Francisco, bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería e Industrias; y que dicho trabajo cumple con las condiciones requeridas para ser sometido a la presentación pública y evaluación por parte del Jurado examinador que se designe.

Bonilla Venegas Félix Vladimir, PhD.

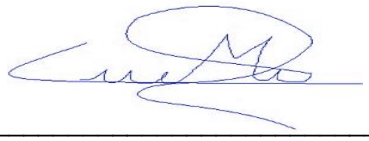
**DIRECTOR DEL TRABAJO**

C.I. 1710300045

# DECLARACION JURAMENTADA DEL AUTOR

Yo, César Francisco Moya Guerra, portador de la cédula de identidad N.º 1719000125, declaro que el trabajo aquí descrito es de mi autoría, que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en ese documento.

La Universidad UTE puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.



f:_____

MOYA GUERRA CÉSAR FRANCISCO

1719000125

# DEDICATORIA

Quiero dedicar este trabajo de titulación a mis padres quienes fueron indispensables para realizar este logro en mi vida, su apoyo incondicional que siempre me brindaron para nunca rendirme y realizando sacrificios tanto económicos como morales para que siempre siga y logre mis sueños.

También agradezco a mis hermanas quienes fueron pilares importantes en mi vida para lograr todas mis metas, dándome consejos y ayudándome en momentos difíciles, permitiéndome formarme de correcta manera tanto a nivel profesional o personal. Ellas me motivaron en cada etapa de mi vida.

# ÍNDICE DE CONTENIDOS

**PÁGINA**

# ÍNDICE DE TABLAS

**PÁGINA**

# ÍNDICE DE FIGURAS

**PÁGINA**

# ÍNDICE DE ANEXOS

**PÁGINA**

# RESUMEN

El presente proyecto propone el diseño de un controlador neuronal de fuerza para un exoesqueleto de extremidad inferior. En este proyecto se aplicó el modelo en V para el diseño del sistema de control para sistemas Mecatrónicos. En la etapa inicial de este proyecto se definió los requerimientos del sistema de control, los cuales delimitan los parámetros de la estructura de este. En estos requerimientos se marcan el lenguaje de programación en C/C++ y las placas de desarrollo a usarse en el avance del proyecto. Posteriormente, se realizó la definición de las funciones específicas de un control con red neuronal artificial previamente entrenada, donde se determinó el funcionamiento de cada etapa del entorno de simulación el cual se encontraba en el software Matlab-Simulink, así como el tipo de dato de entrada el cual son señales electromiográficas que son generadas por los músculos del ser humano. Con esto se implementó un diseño de alto nivel, definiendo e implementando cada bloque de control del entorno de Simulink necesario para el sistema, el cual se encarga de procesar las señales que entra al sistema. Todo se realiza para la implementación del sistema de control en las diferentes placas de control, esto a razón de poseer distintos tipos de estructura en su arquitectura y en específico de tener implementado diferentes microcontroladores. Esto por motivo de observar y analizar el comportamiento de cada microcontrolador implementado en la placa de control, y dar a conocer el mejor comportamiento para entender el escenario ideal a usar cada una de estas placas. Todo esto se realizó con el método PIL, el cual permite generar código programable para distintas placas de desarrollo. Para la etapa final, se realiza una observación y análisis profundo de cada comportamiento, tomando registro posterior al procesamiento de diferentes señales electromiográficas.

**Palabras Claves:** Placa de desarrollo, PIL, red neuronal artificial, C/C++, señales electromiográficas, procesador, entorno de control, software Matlab-Simulink, tiempo.

# ABSTRACT

The present project proposes the design of a neural force controller for a lower extremity exoskeleton. In this project, the V-model was applied for the design of the control system for Mechatronic systems. In the initial stage of this project, the requirements of the control system were defined, which delimit the parameters of its structure. These requirements mark the programming language in C / C + + and the development boards to be used in the progress of the project. Subsequently, the definition of the specific functions of a control with previously trained artificial neural network was carried out, where the operation of each stage of the simulation environment which was in the Matlab-Simulink software was determined, as well as the type of input data which are electromyographic signals that are generated by the muscles of the human being. With this, a high-level design was implemented, defining, and implementing each control block of the Simulink environment necessary for the system, which is responsible for processing the signals that enter the system. Everything is done for the implementation of the control system in the different control boards, this because of having different types of structure in its architecture and specifically having implemented different microcontrollers. This is for the purpose of observing and analyzing the behavior of each microcontroller implemented in the control board, and to publicize the best behavior to understand the ideal scenario to use each of these boards. All this was done with the PIL method, which allows to generate programmable code for different development boards. For the final stage, an in-depth observation and analysis of each behavior is carried out, taking record after the processing of different electromyographic signals.



**Keywords:** Development board, PIL, artificial neural network, C / C++, electromyographic signals, processor, control environment, Matlab-Simulink software, time.

# 1. INTRODUCCIÓN

El avance tecnológico en los últimos años ha causado la incorporación de la robótica en el ser humano, esto a razón de la búsqueda constante de mejora en la eficacia y eficiencia de las actividades tanto personales como laborales.

Uno de los resultados de este avance se plasma con la creación del exoesqueleto, el cual ha ido evolucionando en el transcurso del tiempo desde su aparición a mediados del siglo anterior. (Alberto López Delis, Andrés F. Ruiz Olaya, 2012) Este invento se convirtió en una herramienta que está en constante desarrollo para facilitar una mejor calidad de vida del usuario.

Este dispositivo es un traje robótico usado como una vestimenta diaria el cual puede ser pasivo para el soporte o activo para el aumento de movimientos diarios que el ser humano realiza constantemente. Dependiendo del formato del diseño este tendrá o no actuadores que darán asistencia al usuario. (Manuel Alejandro Chávez Cardona, 2011)

Los usos más comunes que se dan a estos son: la rehabilitación y asistencia a individuos con lesiones, y el aumento de la fuerza, velocidad y resistencia física, para el cual existen exoesqueletos enfocados únicamente en la extremidad inferior del cuerpo humano.

Sus controladores dependen de los movimientos y acciones que se vayan a realizar, esto debido a que solo la configuración activa posee actuadores a controlar para el aumento o rehabilitación física del usuario. Existen diversos tipos de control, uno de ellos es el control inteligente mediante redes neuronales, basado en distinguir diferentes movimientos a través de sensores ubicados a lo largo de las extremidades tanto superior como inferior. Estas señales son variaciones eléctricas que son producidas al momento de realizar accionamiento en el miembro. (Ranko Zotovic Stanisic, 2014)

Estos controles se encuentran enfocados en dar un soporte al usuario con una integración de software y hardware de la herramienta, para lo cual es necesario un estudio de cada variable posible que se pueda generar tanto como parte del usuario, como del entorno en el que se vaya a utilizar esta herramienta, para el aumento de su eficiencia. (Cuesta, 2019) El controlador más común de un exoesqueleto es híbrido dividido en tres etapas, la primera encargada de la velocidad de los cambios por lo que es usado un tipo de control ON/OFF, el segundo es basado en máquinas de estado encargándose de gestionar el límite de accionamiento de los actuadores dependiendo de la contracción muscular, así como el lugar y sentido de movimiento.

Por último, el control inteligente es el encargado de procesar la información recibida de las Señales Electromiográficas, también conocidas como EMG, para decidir la mejor movilidad posible del musculo. Estas señales son impulsos eléctricos generados por el movimiento de fibras musculares, los

cuales tienen una pequeña variación en cada movimiento causando una dificultad en el muestreo de cada uno. Son mayormente utilizadas en la medicina por su método de lectura no invasivo, dando una mayor comodidad al paciente. (Gustavo Aguirre, Ángel Flores, Noé Alba, Carlos Acosta, Ismael Canales, 2015)

Para la implementación de este tipo de controladores se utilizan cajas de control, las cuales deben estar adaptadas para soportar alteraciones físicas del ambiente para el resguardo de la electrónica encargada de ejecutar las acciones del controlador. Por lo general se utiliza un microprocesador, encargado de ordenar y evaluar las acciones que el exoesqueleto deba realizar dependiendo de las órdenes del usuario. (Franco Núñez, 2017)

Un ejemplo de estos dispositivos es el exoesqueleto asistencial hibrido “HAL”, el cual se observa en la Figura 1. Este es desarrollado por la colaboración de la universidad japonesa *Tsukuba* y la empresa del mismo país *Cyberdyme Systems* en el año 2011, el cual fue creado en primera instancia para la asistencia de actividades diarias con la ayuda del aumento de fuerza física, posteriormente al realizar observaciones fue cambiado su uso siendo orientado a la rehabilitación de personas con lesiones en la medula espinal.

**Figura 1**. Diseño de exoesqueleto HAL.
(Paniagua, 2018)

La estructura del exoesqueleto HAL está formado por 6 articulaciones, 4 articulaciones activas que se encuentran ubicados dos en la cadera y una en la rodilla de cada pierna, generando la necesidad de un control de fuerza en cada uno de ellos, y 2 articulaciones pasivas que se encuentran ubicadas en cada uno de los tobillos, dando un soporte a toda la estructura mediante un circuito cerrado de sensores de presión colocados en la superficie del pie.

El controlador de dicho exoesqueleto es la conjugación de una integración funcional con un intercambio neuronal motora, es decir que el funcionamiento de este dispositivo es diferente a cualquier otro, dado que otros exoesqueletos funcionan con mandos externos y el exoesqueleto HAL funciona a través de señales generadas por los pacientes. (J.M. Grosso, 2010)

Este tipo de control es específico para la rehabilitación clínica de pacientes con algún problema físico, a razón de esto varía dependiendo del caso. Estas variaciones son conocidas como: *Control Voluntario Cibernético* y *Control Autónomo Cibernético*, cada una con una especificación distinta para la mayor eficacia del tratamiento del paciente.

Todo este sistema está dividido en 4 secciones las cuales son: actuadores, sensores, hardware de seguridad, PC de visualización de datos y microcontrolador embebido con software de control, este último recibe todos los datos de señales realizadas por el usuario mediante una red SPI, la cual se enfoca en una lectura y escritura rápida de los sensores y actuadores para una respuesta temprana como se observa en la Figura 2.

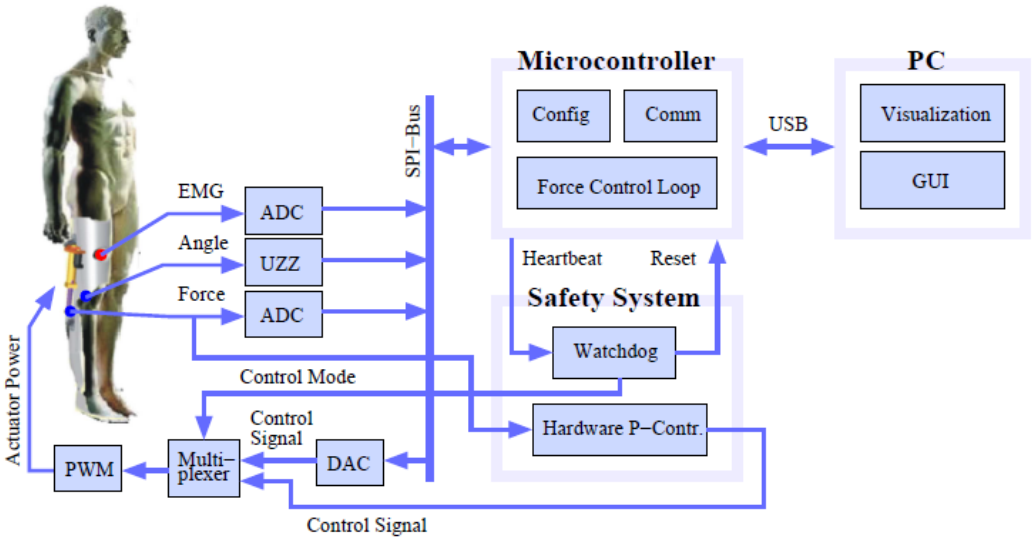



**Figura 2**. Hardware de “El Sistema de Control”.
(Christian Fleischer, Gunter Hommel, 2010)

Este procesador es el Atmel Mega 32, encargado de realizar el procesamiento de control del exoesqueleto, por su facilidad de adquisición y manipulación de diversas variables. Este es un procesador muy usado en elementos electrónicos dadas características que posee, las cuales corresponden a: capacidad de procesamiento de hasta 16 MHz, memoria flash de 32 Kb programables, contadores de 8 bits, entre otros.

Estas características son perfectas para un trabajo sincronizado con redes neuronales, las cuales necesitan unificar la lectura y la escritura del exoesqueleto para una mayor eficiencia en el campo de aplicación en la rehabilitación. (Daniel Peralta, Andrés Herrera-Poyatos, Francisco Herrera, 2016)

Otro exoesqueleto muy usado es el EksoVest, el cual se ocupa en las industrias de fabricación automotriz FORD para reducir el desgaste físico, lesiones y malestares de los operarios. Este se enfoca en las extremidades superiores para disminuir la fatiga de los usuarios, por las extensas horas de trabajo pesado al cual están sometidos. (Aurélei Moyon, Emilie Poirson, Jean-François Petiot, 2018)

El EksoVest observado en la Figura 3, es un exoesqueleto pasivo, lo que significa que se enfoca en dar soporte a la carga de actividades repetitivas diarias para la reducción de lesiones laborales. Diseñada para personas con altura entre los 152 cm y los 193 cm de alto, con actuadores pasivos “resorte” para añadir una fuerza de entre 2.2 y 6.8 Kg en las extremidades superiores.


**Figura 3.** Exoesqueleto EksoVest.
(Alexis Gonzalo, Leopoldo Javier, 2018)

Estas especificaciones dan la facilidad de tener una ausencia de sistema de control, a razón de no necesitarlas por su diseño mecánico dando una mayor facilidad de adquisición en el mercado por su facilidad de fabricación. (Kelson. Denean M., Kim, Sunwook, Nussbaum, Maury A., Srinivasan, Divya, 2019)

El presente proyecto propone la generación de un sistema embebido de control debido a que el uso de estos sistemas genera una mayor eficacia al momento de agrupar y comprobar una gran cantidad de información en el controlador, dando de esta forma una mejor respuesta del exoesqueleto tanto en tiempo como en acciones realizadas.

En la actualidad los sistemas de control de los exoesqueletos son simples y en su mayoría son estrategias ON/OFF, estas estrategias producen oscilaciones en los eslabones del exoesqueleto que son transmitidos al usuario, causando que el control no sea eficaz y eficiente para el usuario, causando una incertidumbre de este al momento de utilizar el exoesqueleto.

La oscilación se genera a causa de la imposible estimación del punto de referencia dado por el usuario, es decir, el exoesqueleto no puede detenerse en la posición que desea el usuario, así como la no respuesta inmediata del control al momento de llegar al lugar deseado (Raíz, 2019).

Por tales razones no se ha podido implementar a gran escala el exoesqueleto en el mercado, debido a la respuesta no inmediata de estos. Esto generado por la falta de conocimiento del público, sobre los beneficios posibles del control neuronal de un exoesqueleto de aumento de fuerza.

Se pretende cambiar esta incertidumbre por parte de los usuarios con una implementación de control más adecuado a las acciones a realizar. Mediante el control neuronal que asemeja el pensamiento humano, se utilizará una mejor respuesta a las acciones solicitadas por el usuario para una aproximación al valor deseado más exacto. Esto con ayuda de una caja de control embebido con diversas variaciones de microcontroladores.

Los sistemas de control inteligente realizan aproximaciones a las acciones realizadas por parte del ser humano. La red neuronal artificial dará un reflejo semejante a los pensamientos realizados por el usuario, es decir, se realizará la acción de movimiento de una manera mejor controlada eliminando la oscilación posible al momento de llegar al punto solicitado. También se ejecutará la aproximación adecuada para la compensación de fuerza al momento de realizar acciones con sobrecarga, de forma que la fuerza que ejercen los actuadores sea proporcional a la necesidad que el músculo requiera asistencia. (Daniel Peralta, Andrés Herrera-Poyatos, Francisco Herrera, 2016)

Con base a lo anteriormente expuesto se definió el siguiente objetivo general:

Diseñar un sistema de control mediante redes neuronales artificiales para un exoesqueleto de aumento de fuerza de extremidad inferior.

Para llevar a cabo el objetivo general, se planteó los siguientes objetivos específicos:

- Determinar la arquitectura de la red neuronal artificial del sistema de control.
- Seleccionar el hardware para la implementación del sistema de control mediante redes neuronales artificiales.
- Desarrollar el código embebido de las redes neuronales.
- Validar el sistema de control mediante pruebas de HIL/SIL-MIL.

El sistema de control que controla varia en configuración para los diferentes tipos de placas de desarrollo, esto a razón de diferentes tiempos de reloj para el procesamiento de señal, así como distintos campos posibles de uso.

El sistema de control se desarrolló en lenguaje de programación C++ y utilizando como hardware diversa configuración de placas de desarrollo como: Arduino Mega, Raspberry Pi 3B+ y Arduino Due.

Estos sistemas vienen fabricados con la más eficiente arquitectura posible para el máximo aprovechamiento de los microcontroladores. Cada uno con características únicas para diferentes tipos de control, y con diferentes valores de mercado para la accesibilidad de usuarios variados con necesidades diversas.

El administrador del sistema puede modificar la programación del sistema de control en cada uno de los microcontroladores y capturar las señales electromiografías “EMG” para su posterior procesamiento mediante simulación en Matlab-Simulink.

# 2. METODOLOGÍA

Para el desarrollo de este proyecto se aplicó el modelo en V para el diseño del sistema de control para sistemas Mecatrónicos, como se observa en la Figura 4.

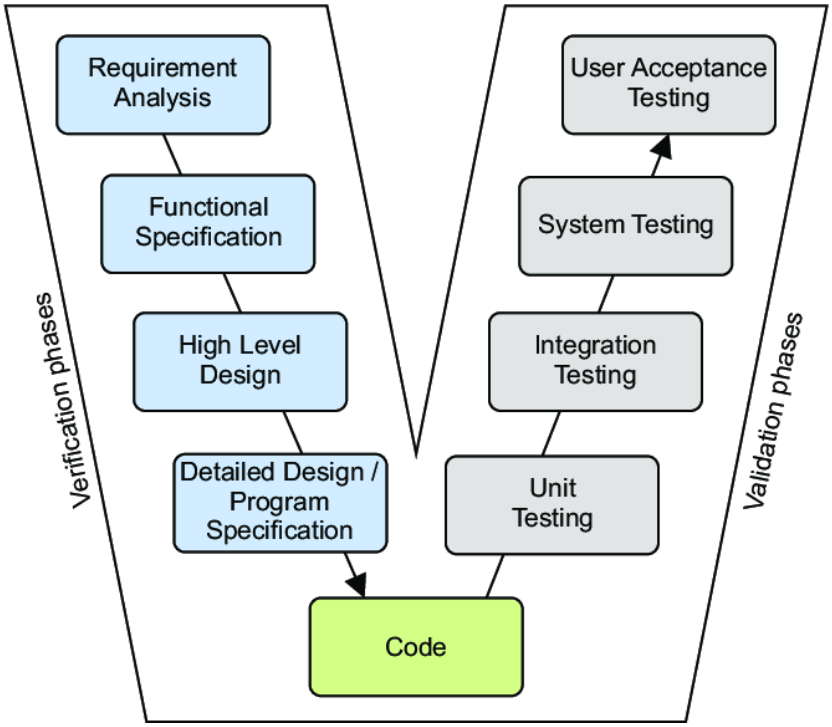



**Figura 4**. Diseño de Sistema de Control por Modelo-V.
(Jiri Koziorek, Jaromir Konecny, 2018)

Con base al diseño de control se realizó el protocolo de programación embebido para diferentes tipos de placas para seleccionar la mejor opción entre beneficio y precio que se pueda encontrar en el mercado. Estas placas poseen diferentes características que varían tanto el tiempo de muestreo como la capacidad de procesamiento en la memoria de cada una, dando datos similares y diferentes con respecto al tiempo.

El diseño del sistema embebido se realizó modificando el programa base de simulación en computadora, tomando en cuenta cada característica especifica de las placas para conocer y compara el costo beneficio de cada una y elegir la más adecuada para esto. Todo esto tomando en cuenta el tiempo de muestreo, la rapidez de respuesta en cada señal y el costo de cada una.

Posteriormente, se realizó la etapa de integración de la red neuronal artificial con diversas señales previamente adquiridas para llegar a una conclusión general de las placas y sus microcontroladores.

Finalmente, los sistemas embebidos se sometieron a varias pruebas de estrés para conocer el mejor resultado posible en campo.

## 2.1. REQUERIMIENTOS

En la Figura 5 se observa los requerimientos para el sistema embebido.

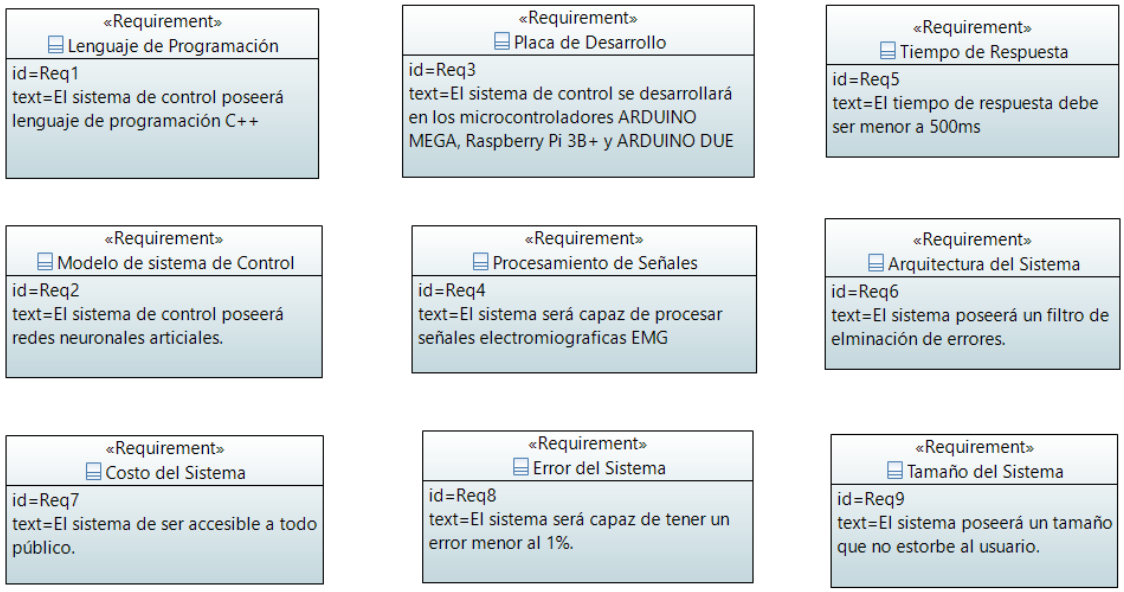



**Figura 5**. Diagrama de requerimientos del método y sistema de control de exoesqueleto de miembro inferior.

El desarrollo del sistema de control inicia con la etapa de requerimientos, los cuales son definidos de la siguiente forma:

- El sistema de control debe estar formado por el lenguaje de programación C++.
- La red neuronal artificial procesará los datos adquiridos con una velocidad menor a 500 ms.
- El sistema de control será diseñado con una estructura general para el uso en las placas Arduino Mega, Arduino Due y Raspberry PI 3+.
- El sistema poseerá un control neuronal que definirá la posición del miembro inferior del usuario.
- La arquitectura del sistema de control está definida para el procesamiento de Señales Electro-Miográficas “EMG”.
- Para la mayor eficiencia el sistema de control poseerá un filtro que ayudará a disminuir el error.
- La accesibilidad del usuario para adquirir la placa de control debe ser para todo público.
- El resultado del sistema debe dar un error menor al 1% con respecto a la simulación por computadora.
- El hardware del sistema debe poseer un tamaño que no ocupe un volumen mayor a 123.75 centímetros cúbicos.

## 2.2. DISEÑO CONCEPTUAL

En la Figura 6 se puede observar el diagrama de la estructura del sistema para el diseño de un controlador neuronal de fuerza para un exoesqueleto de extremidad inferior.

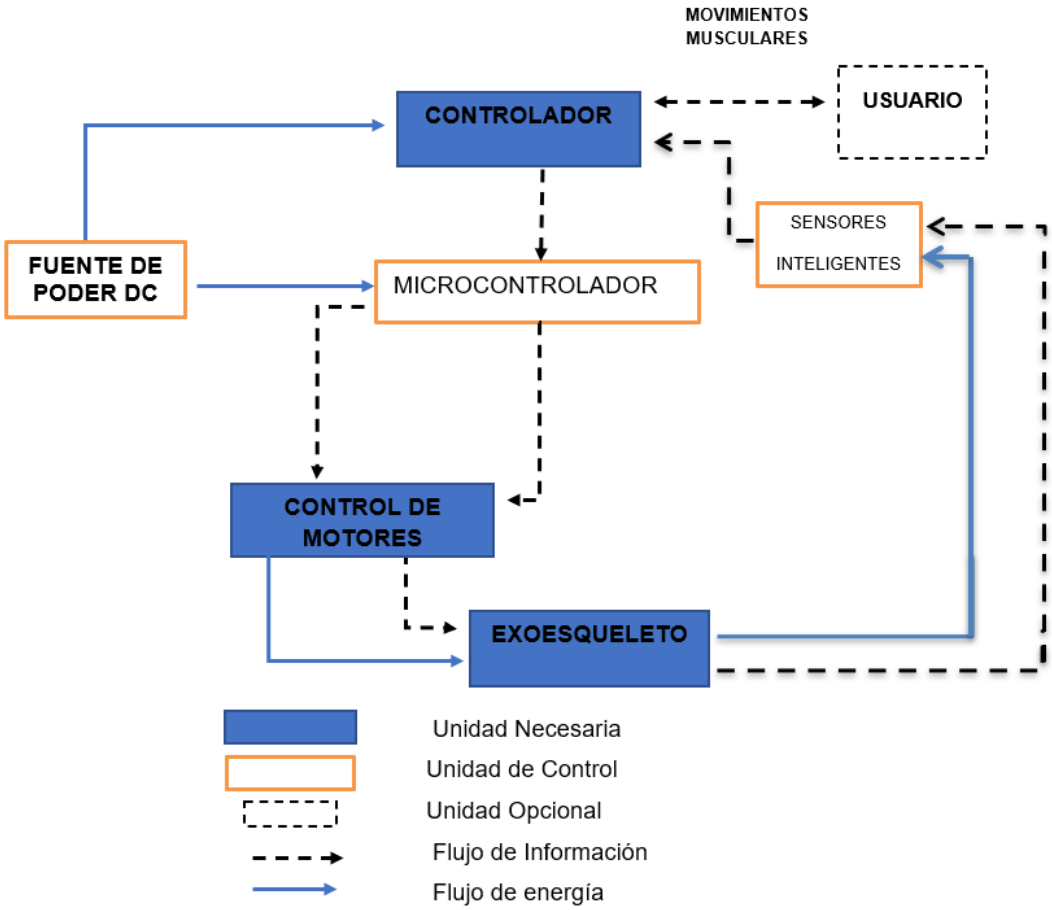



**Figura 6.** Diagrama de la estructura del sistema

En el diagrama que se observa se puede distinguir la lógica de flujo de información y la lógica de energía necesaria para todos los componentes del sistema. La base de la estructura es identificada con el exoesqueleto, el cual emplea el control de la extremidad inferior mediante redes neuronales artificiales y señales Electro-Miográficas tomadas del usuario.

La sección de la fuente de energía se identifica con una conexión directa a la placa de desarrollo por cable, el cual suministra la energía necesaria para el correcto funcionamiento del sistema de control.

## 2.3. FUNCIONALIDAD DEL SISTEMA

Mediante un diagrama de casos se puede representar la funcionalidad del sistema, el cual se observa en la Figura 7.

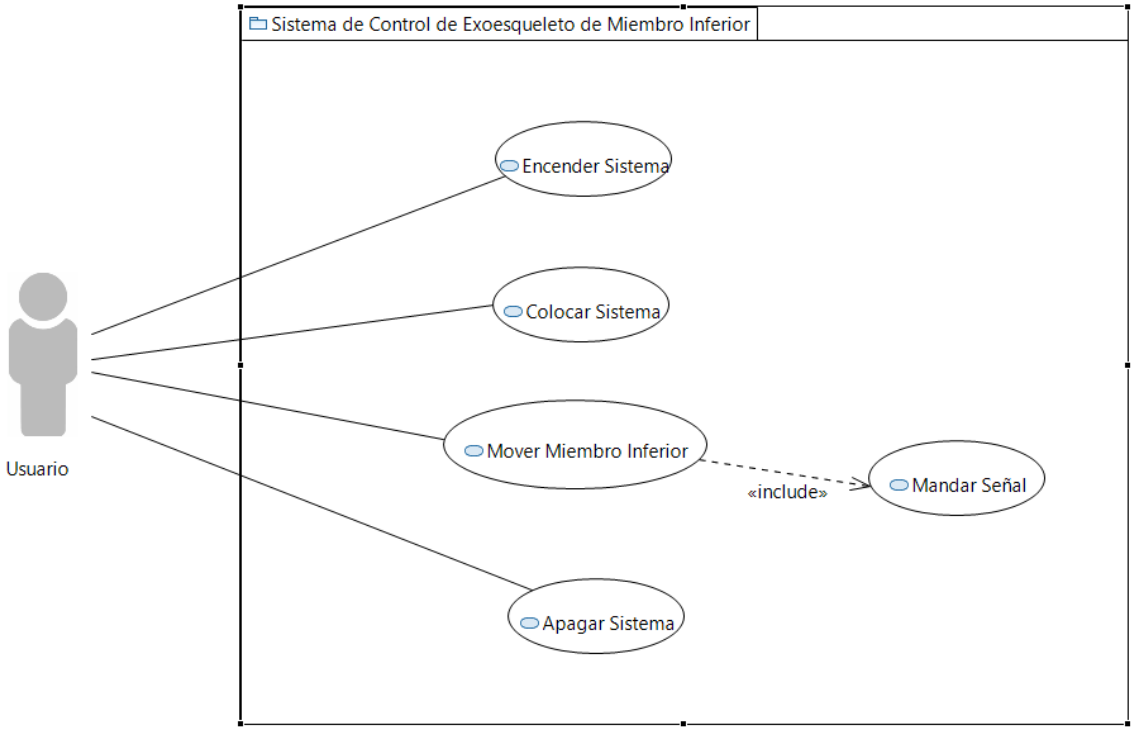



**Figura 7.** Diagrama de caso de uso del sistema de controlador neuronal de fuerza para un exoesqueleto de extremidad inferior.

Por medio de este diagrama se observa la interacción existente entre usuario y sistema, por el cual se puede determinar la forma de uso de este. Como se muestra, el usuario puede encender y apagar el sistema a su antojo por medio de la activación de corriente en el mismo.

## 2.4. ARQUITECTURA DE CONTROL

### 2.4.1. PLACAS DE DESARROLLO

El sistema de control debe ser capaz de funcionar de manera ideal en cualquier tipo de microcontrolador, para esto se tomó 3 diferentes placas de desarrollo con microprocesadores a distintas dimensiones y velocidades de procesamiento.

Las placas de desarrollo empleadas son:

- Arduino Mega observado en la Figura 8, está formado por el microprocesador ATmega2560, el cual posee 54 entradas y salidas

digitales con 15 especializadas para señales PWM. Esta placa de desarrollo trabaja a una frecuencia de reloj con un cristal de 16 MHz, para ejecutar los programas almacenado en su memoria interna con un límite de capacidad en la memoria flash de 256 Kb y un procesamiento de 8 bits. La velocidad de este procesador está determinada por los millones de instrucciones por segundo “MIPS”, los cuales son 1 MIPS por cada MHz dando un total de 16 MIPS.

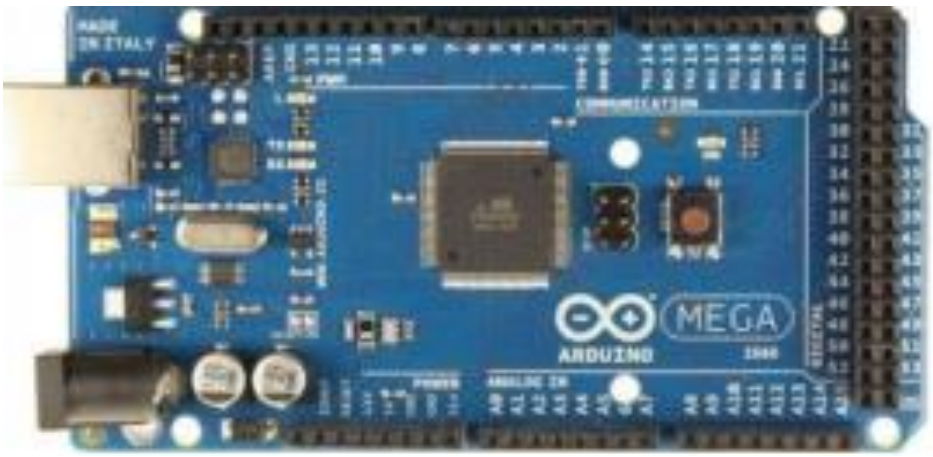



**Figura 8**. Placa de Desarrollo Arduino Mega 2560.
(Akbar iskandar, 2017)

El funcionamiento de procesador debe ser alimentado con un voltaje de entrada de entre 6 a 20 V, el cual puede ser suministrado por cable USB o cable de alimentación continua. Para el almacenamiento se realiza mediante cable USB con ayuda del entorno de desarrollo Arduino IDE, el cual viene preprogramado con el gestor de arranque para el microprocesador. (Veton Këpuska, Humaid Saif Alshamsi, 2016)

- Raspberry Pi 3B+ mostrado en la Figura 9, con procesador Broadcom BCM2837B0 “Cortex-A53” (ARMv8), a un funcionamiento de 64 bits con una frecuencia de reloj de 1.4 GHz. La velocidad de este procesador es directamente proporcional al reloj de la placa, es decir de 1.4 GHz para la mejor eficiencia entre el procesamiento y el estado de la estructura del procesador para que no exista ningún daño con la integridad de este con respecto a la temperatura generada. Este procesador con un diseño de enfriamiento ideal puede llegar a una velocidad de 1.56 GHz dando una mejor respuesta a multifunciones.

**Figura 9.** Placa de Desarrollo Raspberry Pi 3B+.
(Mateusz Sałuch, Daniel Tokarski, Tomasz Grudniewski, Marta Chodyka, 2018)

Este procesador posee la capacidad de 40 pines de conexión y puertos para USB 2.0, cámara, pantalla táctil, toma de auriculares y video, y capacidad de aumento de memoria mediante Micro SD.
La Rasberry Pi 3B+ posee la opción de conectividad de red mediante la Gigabit Ethernet over USB 2.0, la cual permite realizar transferencias de programas y modificaciones de archivos instalados mediante conexión inalámbrica. Esta placa de desarrollo, por todas las características antes mencionadas, permite ser tratada como una CPU aparte el cual posee su propio interfaz y permite una mejor manipulación de este.
A su vez es posible realizar conexiones de datos de la placa a un CPU aparte, el cual permite una mayor disponibilidad de softwares para usuarios menos experimentados y para la realización de experimentos de diferente índole, como se puede observar en la Figura 10.

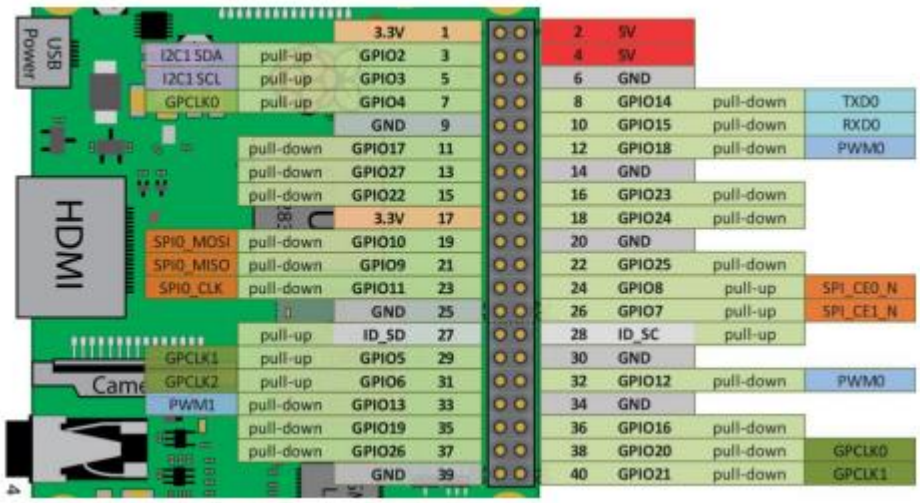



**Figura 10.** Descripción de pines GPIO Raspberry Pi 3 B+.
(Mateusz Sałuch, Daniel Tokarski, Tomasz Grudniewski, Marta Chodyka, 2018)

- Arduino Due observado en la Figura 11, está formado por el microcontrolador Atmel SAM3X8E convirtiéndola en la primera placa de desarrollo de la empresa Arduino con base en ARM, dando la característica especial de poseer en el núcleo el procesamiento Cortex-M3 de 32 bit a diferencia de las placas comunes de la empresa Arduino que poseen una MCU de 8 bit.
  Esta placa de desarrollo posee 54 pines digitales, de los cuales 12 están diseñadas para salidas PWM, además contiene 12 pines analógicos y 4 UARTs. Otra diferencia que contiene el Arduino Due a comparación de las demás placas de desarrollo de Arduino, es la fuente de alimentación que solo necesita 3.3 V en lugar de los 5 V comunes.

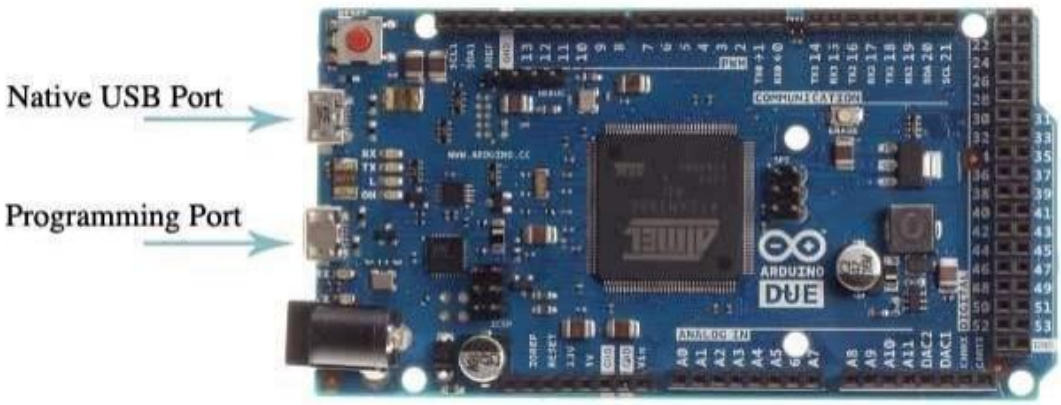



**Figura 11.** Placa de Desarrollo Arduino Due.
(Arduino, 2017)

Este microprocesador trabaja a una velocidad de 84MHz con un oscilador de reloj interno lento de 32.768 Hz en modo de derivación y un oscilador en cristal o cerámica de 3 a 20 MHz. A su vez posee un oscilador RC para las salidas que varían entre 4, 8 o 12 MHZ y un oscilador de 96 a 192 MHz para las entradas, y un control de alta velocidad para USB de 480 MHz una memoria de 96 K, separados en 2 bancos de memoria de 64 Kb y 1 banco de memoria de 32 Kb.
La placa de desarrollo posee 2 relojes de cristal de 16 y 12 MHz, para su programación posee 2 micro USB, los cuales se diferencian en el estilo de comunicación con la placa puesto que el primer puerto llamado “ProgrammingPort” se conecta con la placa entera para usuarios menos experimentados y la segunda llamada “NativeUSBPort” se conecta directamente con el microcontrolador dando la posibilidad de cambiar la configuración de programación y su arranque. (Sergio Barrachina, Germán Fabregat, José Vicente Martí, 2015)

Estas 3 placas se logran analizar de mejor manera en la Tabla 1, denotando la información más relevante de cada una.

**Tabla 1.** Comparación de placas de desarrollo

| **Placas de Desarrollo** | | | |
|---|---|---|---|
| | **Procesador** | **Velocidad de Procesamiento** | **Costo de Adquisición** |
| **Arduino Mega 2560** | ATmega2560 | 16 MHz | $38,00 |
| **Arduino Due** | Atmel SAM3X8E | 84MHz | $40.30 |
| **Raspberry Pi 3 B+** | Cortex-A53 (ARMv8) | 1.4 GHz | $69,00 |

### 2.4.2. SISTEMA DE GENERACIÓN DE SEÑALES DE CONTROL

Tomando una red neuronal artificial generada anteriormente se realizó el proceso de embebido para las diferentes placas de desarrollo, este comenzaba con el procesamiento de la señal electromiográfica mediante análisis en el dominio del tiempo con ayuda del software Matlab-Simulink.

Para este análisis se toma en cuenta las siguientes características:

$X_i$ representa la señal EMG en un segmento i.

N se refiere a la longitud de la señal EMG.

- **Integrado EMG (IEMG).** Esta función es utilizada como índice para la detección del inicio de la actividad muscular. Su función es una relación con la variación de la secuencia en las señales electromiográficas del usuario. La Ecuación 1 está dada por la sumatoria de los valores absolutos de la amplitud de la señal EMG.

$$\mathbf{IEMG} = \sum_{i=1}^{N} |X_i| \qquad [1]$$

- **Media cuadrática (RMS).** Esta Ecuación 2 enlaza la contracción muscular no fatigante con la fuerza constante, es decir la desviación estándar de estos.

$$\mathbf{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} X_i^2} \qquad [2]$$

- **Varianza (VAR).** Es una ecuación matemática del valor medio cuadrado de la desviación estándar. Esta maneja la señal electromiográfica superficial del musculo como propiedad y es expresada en la Ecuación 3. (Bonilla Félix Vladimir, Moya Marcelo Javier, Anatoly Vitalyevich Evgeny, Anatolevich Lukyanov, Marín Leonardo Emanuel, 2018)

$$\mathbf{VAR} = \frac{\mathbf{1}}{\boldsymbol{N-1}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{X_i - M})^{\mathbf{2}} \quad [3]$$

Donde:

- M es el valor medio de la señal electromiográfica.

Estas características son utilizadas para el análisis en un entorno virtual a fin de entrenar y ejecutar la red neuronal artificial, lo cual permite una validación de datos de salida dando una referencia de procesamiento. Dicho análisis al ser efectuada en el dominio del tiempo da una facilidad en el cálculo en el entorno de desarrollo, permitiendo una representación y observación del sistema de control.

Al obtener los valores de estas características se procedió a tomar los máximos valores de cada una para ejecutar y evaluar la red neuronal artificial. En la ejecución y simulación de este sistema se procede por etapas para el tratamiento de la señal a evaluar, como se observa en la Figura 12.

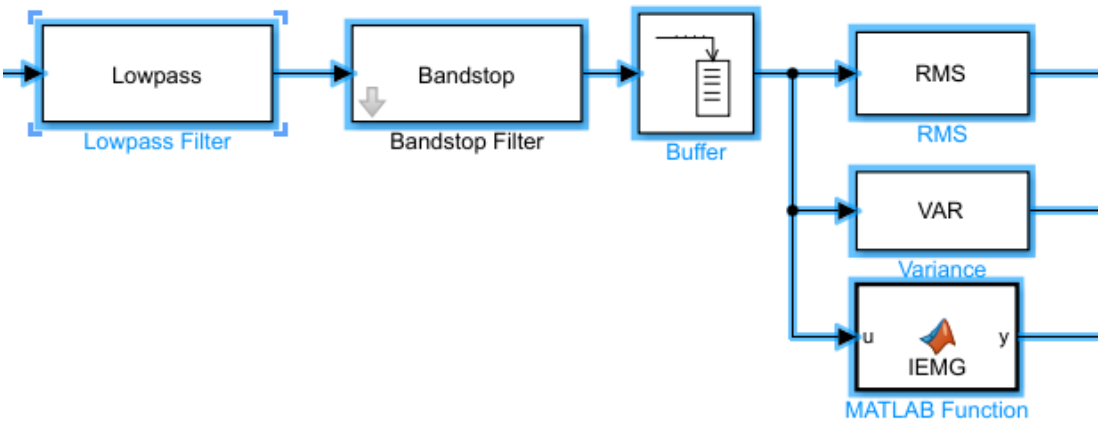



**Figura 12.** Etapas de tratamiento de señal.

El primer paso por realizar es el filtro pasa bajas, el cual permite filtrar las señales de entrada dando paso solamente a señales de baja frecuencia con respecto a la frecuencia de corte. Esta característica es implementada en Simulink con el bloque “Lowpass”, observado en la Figura 13, con un control

del filtro en el parámetro que se configura en IIR por sus siglas en ingles que hace referencia a la respuesta infinita al impulso (Infinite Impulse Response), o FIR por sus siglas en ingles a la respuesta finita al impulso (Finite Impulse Response).

Este bloque está en su configuración IIR, el cual a diferencia del parámetro FIR que realiza cálculos con base a los valores actuales y previos de la entrada, y da a obtener una salida que desaparecerá en el transcurso del tiempo a razón de inexistencia de valores de entrada. Dicha configuración permite tener un dominio tanto de señales de entrada presentes y previos, como de valores previos de su propia salida dando el beneficio de eliminar la inexistencia de valores de entrada para el cálculo realizado. (Cuevas-Jiménez E., Oliva-Navarro D.A., 2013)


**Figura 13.** Bloque de pasa bajos de Simulink.
(Matlab, 2015)

Dicho filtro de las señales de impulso es representado con una función de transferencia la cual se expresa en la Ecuación 4.

$$\frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_r z^{-r}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_c z^{-c}} \qquad [4]$$

Donde:

- $Y(z)$ es la salida del filtro.
- $X(z)$ es la entrada del filtro.
- $a, b$, son los valores correspondientes al vector de parámetros.
- $z, c,$ hacen referencia a las dimensiones del vector.

Luego de haber realizado dicho paso, la simulación procede con un filtro de parada con la implementación del bloque filtro de parada (Bandstop Filter). Dicha característica consiste en repeler las señales de frecuencia que no pertenezcan al grupo determinado por los dos puntos de corte configurados.

En esta etapa las señales de frecuencia de entrada pasan por la configuración IIR, lo cual con ayuda de los límites de frecuencia configurados se logra una señal más limpia permitiendo un mínimo de error por alteraciones externas al sistema de control. A su vez, este bloque posibilita una generación de código más estable en lenguaje C/C++, dando el beneficio de poseer un filtrado de

señal lo más adecuada posible resultando en valores dobles para la ejecución de la red neuronal artificial. (Matlab, MathWorks, 2021)

Siguiendo con la simulación, se llega al bloque Buffer mostrado en la Figura 14, que está caracterizado por realizar procedimiento basado en tramas. Este redistribuye los datos de la señal de entrada de forma separada por cada columna para una producción de datos con un tamaño de contexto diferente, lo cual al configurarlo con valores muy grandes tiende a disminuir su velocidad de salida.

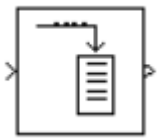


**Figura 14.** Bloque Buffer de Simulink.
(Matlab, MathWorks, 2021)

Esto ayuda a la red neuronal artificial, ordenando los datos de la señal de impulso de forma que se divida en grupos más fáciles de tratar a lo largo del tiempo. Con esto se puede tratar el sistema de una manera más eficiente para la ejecución del programa.

Al tener los datos divididos en diferentes grupos se prosigue con las características de RMS, VAR y el valor medio de la señal electromiográfica, para lo cual se usa los bloques RMS, VAR y TRANSFER FUNTION. Estos permiten realizar las operaciones matemáticas necesarias para el tratamiento de la señal electromiográfica, dando los valores necesarios para la ejecución de la red neuronal artificial previamente entrenada para producir una salida, la cual se encargará de dar el posicionamiento deseado del exoesqueleto.

Al haber completado el análisis y comprobación del funcionamiento del sistema de control por medio de la simulación en Simulink, se continua a realizar el proceso de bucle “PIL” (Processor in the Loop). Esta característica permite realizar una compilación cruzada del código para una sucesión con la descarga y ejecución del sistema de control en la placa de desarrollo.

Para la realización de este proceso se sigue una sucesión de etapas para la configuración del sistema, de modo que se logre obtener un código en C y C++, permitiendo embeber el sistema de control en el hardware.

## 2.5. DISEÑO DE PROGRAMA

### 2.5.1. GENERACIÓN DE CÓDIGO

Como se mencionó en el punto anterior, se debe realizar un PIL del sistema de control para su ejecución en el hardware. Para realizar dicho proceso se debe seguir los sigues pasos:

1. En primer lugar, se debe realizar la comprobación del sistema en los apartados de la ventada de parámetros de la simulación para que su configuración se encuentre en estado discreto. Para esto se debe ir a la configura del modelo manifestado con el icono de tuerca en la barra de herramientas de Simulink, posteriormente se debe modificar las siguientes configuraciones. (Arias, 2020)

    a) En la sección nombrada “Solver” mostrada en la Figura 15, se debe colocar en el apartado de tipo de dato con “Fixed-step”. Posteriormente en el apartado de “Solver” se debe elegir el método no discreto a utilizar para la generación de código. Por último, en el apartado de “Fixed-step size” se debe colocar el tiempo de muestreo más idóneo para cada placa de desarrollo.

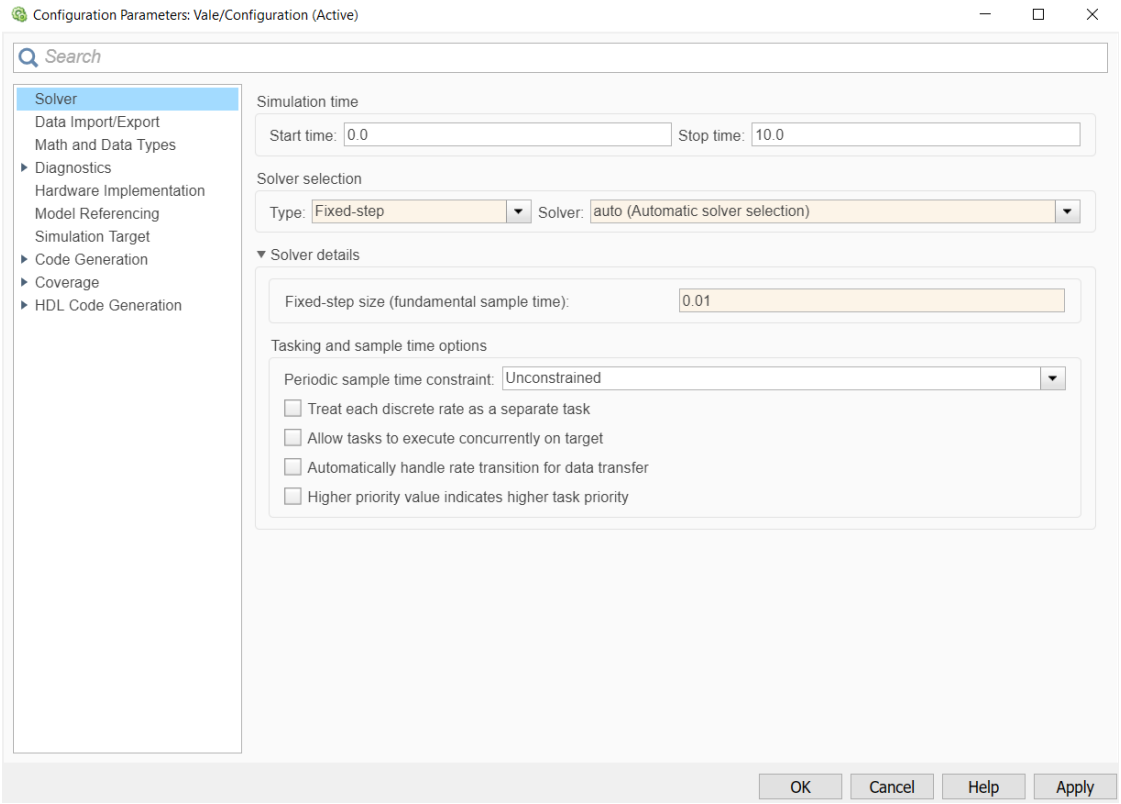



**Figura 15.** Ventana de configuración de “Solver”.

    b) A continuación, en la sección de “Hardware Implementation” en el apartado de “Hardware board” se debe seleccionar la placa

de desarrollo a usar entre Arduino Mega 2560, Raspberry Pi, Arduino Due, como se observa en la Figura 16.

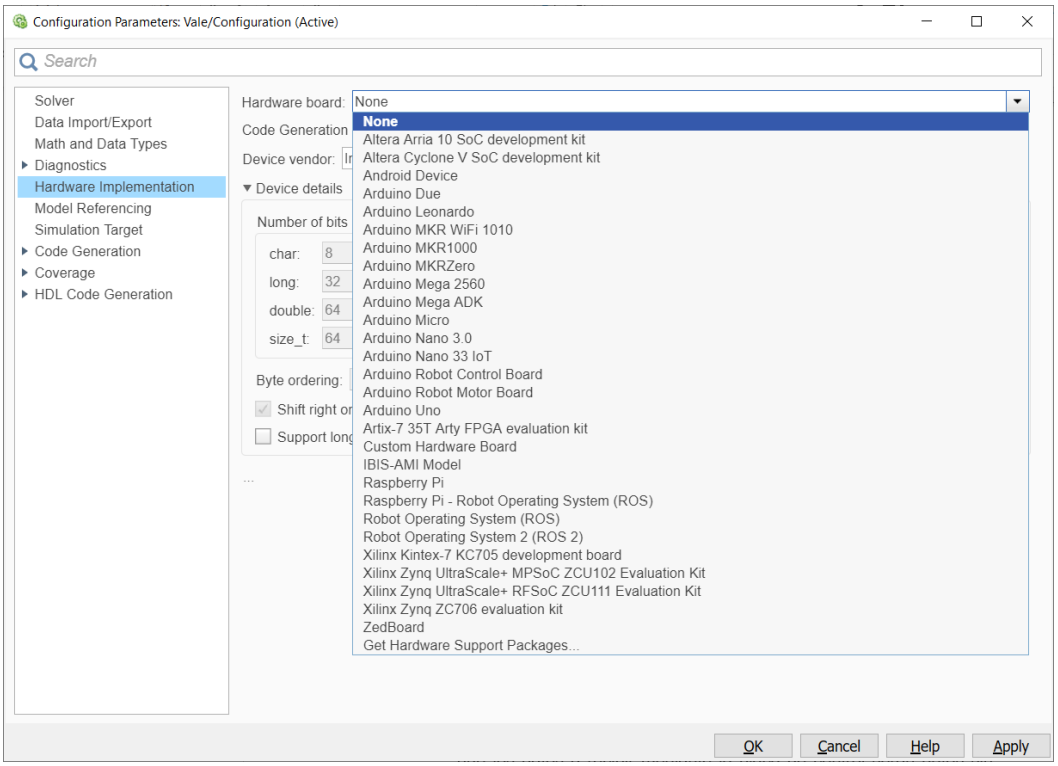



**Figura 16.** Ventana de Configuración de “Hardware Implementation”.

c) Posteriormente en la sección de “Code Generation” indicada en la Figura 17, en el apartado de “Target selection” se debe seleccionar en el botón “Browse” el nombre de “ert.tlc” para la parte de “System target file”, el cual permitirá la generación de código embebido.

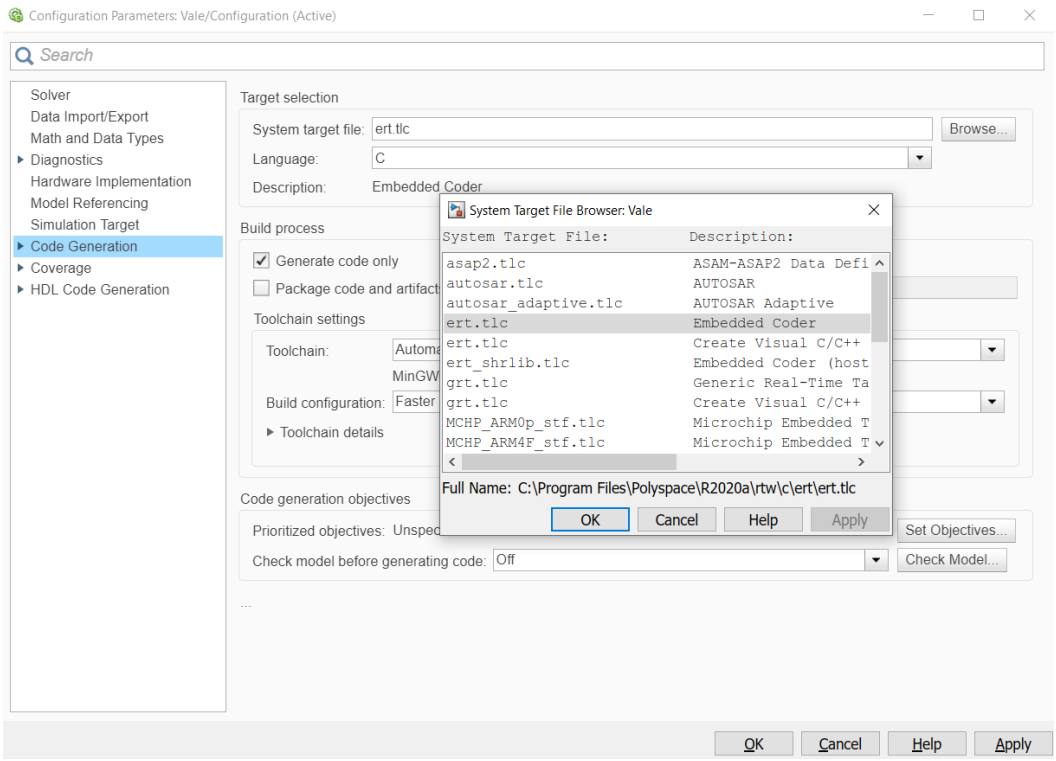



**Figura 17.** Ventana de Configuración de “Code Generation”.

2. Siguiendo el procedimiento para realizar PIL se debe colocar un convertidor de datos que vaya a recibir el sistema de control, a razón que los datos a recibir mediante la placa de control serán datos sin tratar. Para esto se debe agregar dos bloques de Simulink que realizaran la conversión de los datos, el primero es el bloque “Data Type Conversion”, el cual se encarga de transformar los datos de impulso de entrada a datos en formato doble permitiendo el procesamiento adecuado de la placa de desarrollo, a continuación, se agrega el bloque “Gain” permitiendo transformar los datos a escala binaria de base de 10 bits, como se observa en la Figura 18.

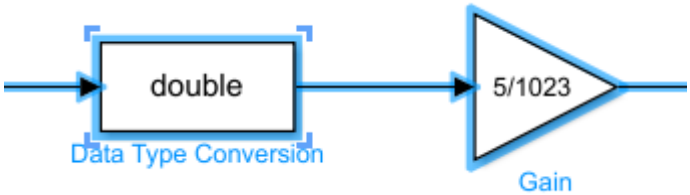



**Figura 18.** Bloques de tratamiento de señal de impulso de entrada.

3. A continuación, en el procedimiento, el sistema de control debe ser situado en un subsistema para la generación de código embebido. Para ello se debe seleccionar todos los elementos deseados del sistema de

control a fin de ubicarlos en el interior de subsistema, y mediante clic derecho en los elementos seleccionados se debe elegir la opción de “Create Subsystem from Selection” para obtener el subsistema deseado indicado en la Figura 19.

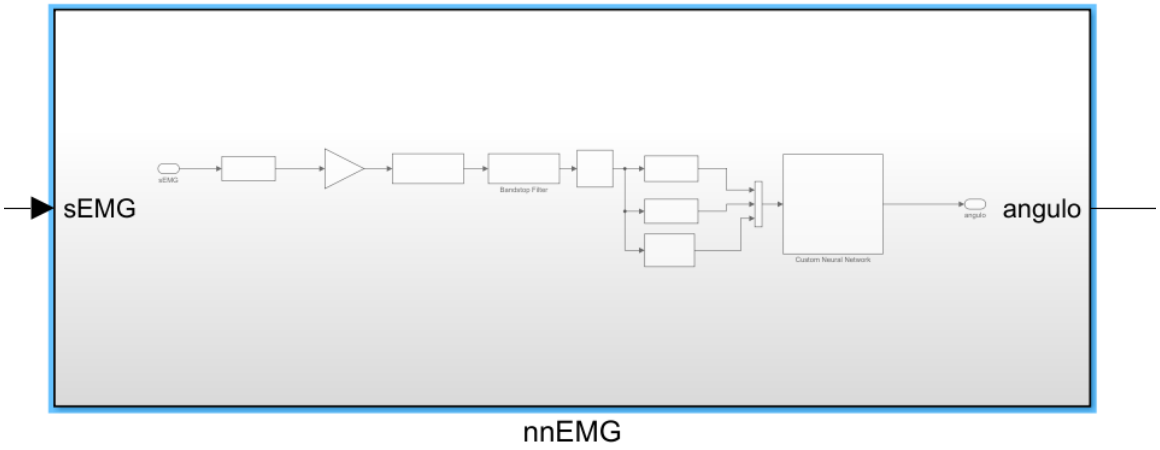



**Figura 19.** Subsistema del Sistema de Control.

Para el último paso del proceso PIL se debe separar en dos secciones, cada una de ellas tendrá diferente procedimiento de montaje sobre las placas de desarrollo. Esto por motivo de las diferentes especificaciones que poseen cada una, a razón que la configuración de productos Arduino son diferentes a la configuración del producto Raspberry Pi.

La primera configuración por realizar se hizo para las placas de desarrollo Arduino Mega 2560 y Arduino Due. En esta se debe seleccionar el subsistema previamente formado, como se muestra en la Figura 20, y dar clic derecho sobre este, esto genera un listado de opciones para elegir en los cuales se seleccionará la sección de “C/C++ CODE” desplegando otra variedad de opciones en donde se escogerá la opción de “Build This Subsystem”.

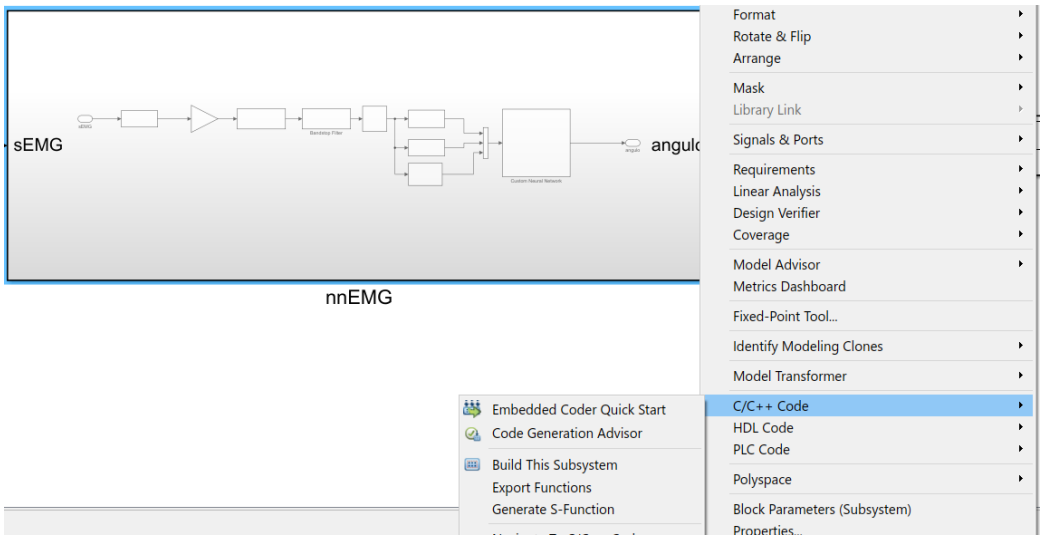



**Figura 20.** Herramientas para el embebido del sistema en placas Arduino.

Posteriormente, se desplegará una ventana indicando el estado del sistema que se desea para generar el código. En esta ventana se debe seleccionar el botón “Build”, el cual dará inicio para la generación del código que se desea.

El tiempo que tomará este paso dependerá de las características del hardware del CPU usado para este procedimiento. Al terminar, se desplegará otra ventana indicando los códigos generados para las placas de desarrollo, los cuales se utilizarán para el desarrollo del programa que se ejecutara en cada una de las placas Arduino, como se observa en la Figura 21.

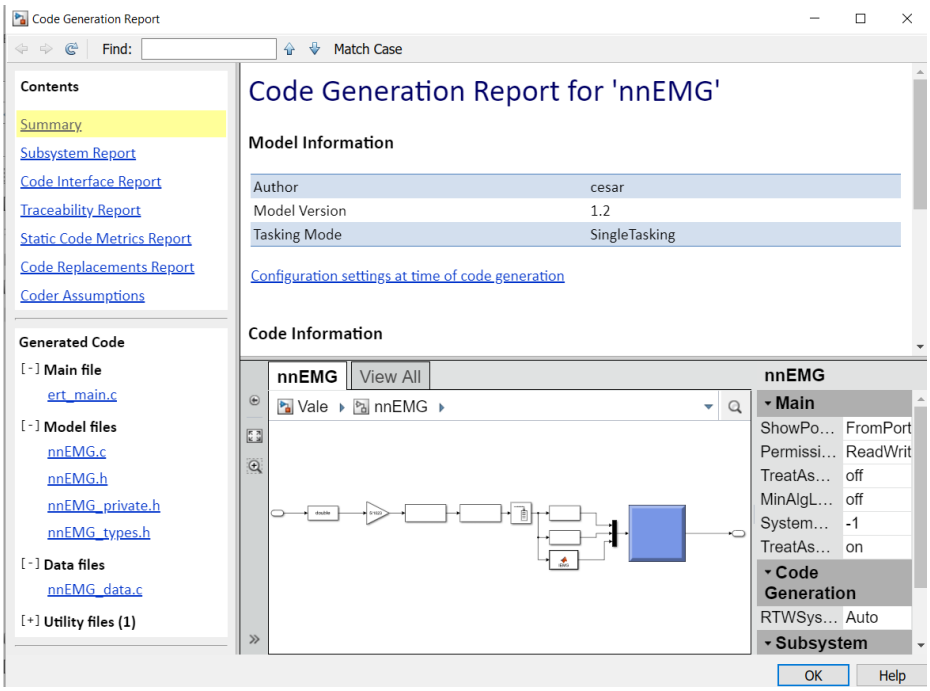



**Figura 21.** Ventana de códigos generados.

En estos códigos obtenidos se destacan la cabeza y cuerpo, los cuales se encargarán de realizar el control de la señal de impulso. Estos se los reconoce por su nombre, el cual está definido por el nombre del subsistema seguido por la extensión específica para cada uno (“nnEMG.h”, “nnEMG.cpp”).

Dicho código generado se lo observa en el Anexo 1, en el cual se denota las librerías utilizadas, así como el proceso que realiza para el sistema de control. Este proceso inicializa con el filtrado de la señal de impulso, posterior a esto se ejecuta observa el tiempo de reacción utilizado, el cual es de 0.128 segundos por cada interacción de la señal.

Con ayuda de estos códigos generados, se desarrolla el programa en el software Arduino para las dos placas de desarrollo. Esto a razón que, a nivel de software, las dos placas pueden actuar del mismo modo en su programación.

Este programa desarrollado es el siguiente:

```
#include "nnEMG.h"

//#include "nnEMG.cpp"

float sEMG,angle;

void setup() {

  Serial.begin(57600);

  nnEMG_initialize();

}

void loop() {

  if(Serial.available()){

    sEMG = Serial.parseFloat();

    nnEMG_U.sEMG = sEMG;

    nnEMG_step();

  }

  angle = nnEMG_Y.angulo;

  Serial.println(angle);

  delay (1);

}
```

Aquí hay que señalar que el programa desarrollado tiene una estructura específica para realizar el control de la señal de impulso. En primer lugar, se denota el llamado de librerías, las cuales son los códigos generados anteriormente, a su vez se define la variable a controlar la cual es la señal de impulso a analizar. A continuación, se determina la velocidad e inicialización del sistema de control.

En el apartado de bucle se define la condición de uso de la señal de impulso y se la dispone como valor de entrada para el sistema de control. Para finalizar se define la salida con el nombre que se encuentra en el sistema de control y denota el valor a obtener de salida. Con las placas de desarrollo de Arduino, la conexión que se debe realizar con el software Matlab se produce con los

bloques de Simulink “To Instrument” y “Query Instrument”, los cuales se muestran en la Figura 22 y se encargan de mandar la señal de impulso al hardware y recibir el valor del ángulo de salida respectivamente.


**Figura 22.** Bloques de Simulink “To Instrument” y “Query Instrument”.

El bloque “To Instrument” se encarga de iniciar el hardware especificado en su configuración por medio de comunicación serial y de inicializar dicho hardware con una velocidad de comunicación deseada. Esto permite enviar datos previamente obtenidos los cuales están guardados en la base de trabajo de Matlab, especificando el tipo de dato a enviar.

Por otro lado, el bloque “Query Instrument” se encarga de recibir los datos procesados del hardware al que está configurado. Esto con la misma configuración del bloque anterior y adicional, con el tiempo de muestreo deseado, se logra consultar los valores de salida que la placa otorgue en cada proceso del bucle.

Todo este proceso de generación y desarrollo de código de control para las placas de desarrollo Arduino se puede evitar en el caso de la placa Raspberry Pi, esto a razón de la existencia del método de embebido directo que posee el software Matlab-Simulink 2020. Este método logra realizar PIL en la placa de desarrollo de forma directa, enviando todo el sistema de control del entorno como la señal de impulso guardad en la base de trabajo de Matlab.

Para dicho procedimiento se debe seleccionar la configuración de hardware exhibida como una tuerca y un PCB enlazados en el apartado de “Hardware” en la barra de herramientas de Simulink 2020. En la ventana desplegada mostrada en la Figura 23, se debe configurar la placa de desarrollo Raspberry Pi en la sección de “Hardware board”.

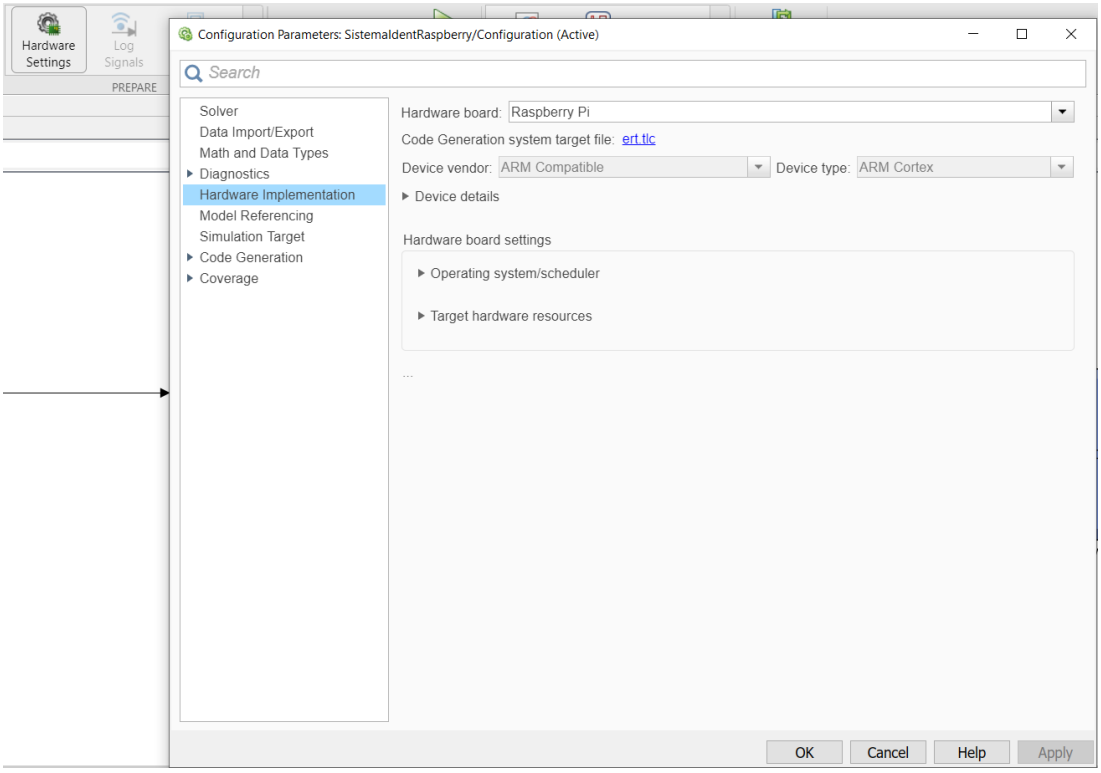



**Figura 23.** Ventana de configuración para placa de desarrollo Raspberry Pi.

A continuación, para la ejecución de la modelo se debe seleccionar la herramienta “Build, Deploy & Start” ubicada en el apartado de “Hardware” de la barra de herramienta indicada en la Figura 24. Esto permite generar, importar e inicializar el modelo de control del sistema directamente en la placa Raspberry Pi, dando una mayor facilidad de PIL con dicha placa de desarrollo.

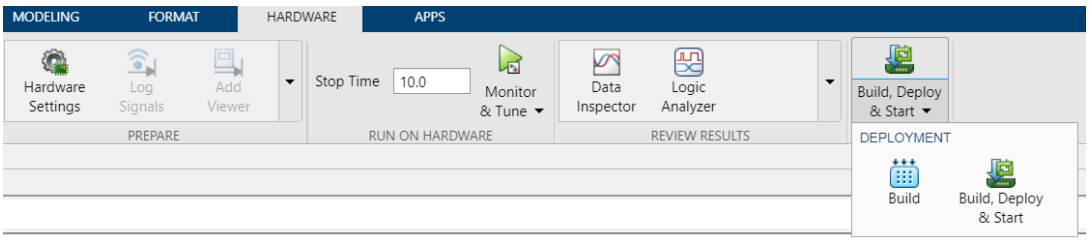



**Figura 24.** Apartado de procedimiento directo PIL.

Posteriormente recibe los datos procesados mediante una lectura directa y los guarda en una variable temporal generada al momento de ejecutar el modelo. Esto permite graficar los valores que recibió y analizar su procesamiento.

# 3. RESULTADOS Y DISCUSIÓN

## 3.1. ANÁLISIS DE PROCESAMIENTO

Luego de realizar PIL en cada una de las placas de desarrollo, se ejecutó el modelo para el análisis y la comparación de las tres diferentes señales de impulso en cada una de estas.

En primer lugar, se debe observar las respuestas simuladas por medio del entorno Simulink, esto para conocer el resultado ideal de cada una de las señales al ser procesadas por la red neuronal artificial. Cada una de las respuestas será tomada como referencia para comprobar el comportamiento de cada una en un tiempo de ejecución de 10 segundos.

La primera señal de impulso se ejecuta con la simulación en el entorno, comparando el ángulo obtenido al ser procesada por la red neuronal artificial en la placa de desarrollo, el ángulo deseado por el usuario guardado en el espacio de trabajo de Matlab y la señal de impulso que se analiza sin ningún tipo de tratamiento.

Como se puede observar en la Figura 25, la red neuronal se aproxima al ángulo deseado por el usuario sin sobrepasar el ángulo máximo permitido por este. Esto quiere decir que la red neuronal artificial está trabajando de manera eficiente en su control, sin realizar movimientos excesivos, los cuales pueden generar lesiones o problemas en el accionamiento del usuario.

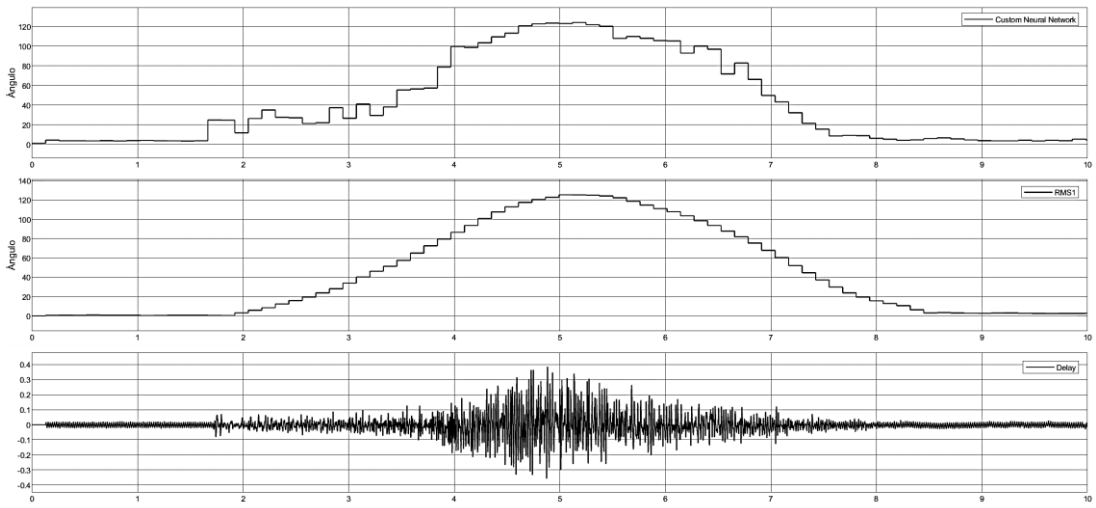



**Figura 25.** Grafica de simulación de señal de impulso 1.

A continuación, se realiza el mismo procedimiento de simulación del sistema con la segunda señal de impulso, para observar y analizar el comportamiento de la red neuronal artificial ante una segunda señal que posee un comportamiento diferente.

Dicho comportamiento se observa en la Figura 26, el cual se diferencia por su tiempo de accionamiento deseado en el ángulo previamente guardo, esto dará

ayuda para el análisis del comportamiento de la red neuronal artificial ante una señal de impulso con menor tiempo de reacción al momento de realizar la inicialización de sistema.

Con esto se logra observar la variación del ángulo obtenido ante situaciones de estrés para la red neuronal artificial. Dicha simulación se enfoca en usuarios con un comportamiento diferente, logrando causar una tensión mayor al sistema de control.

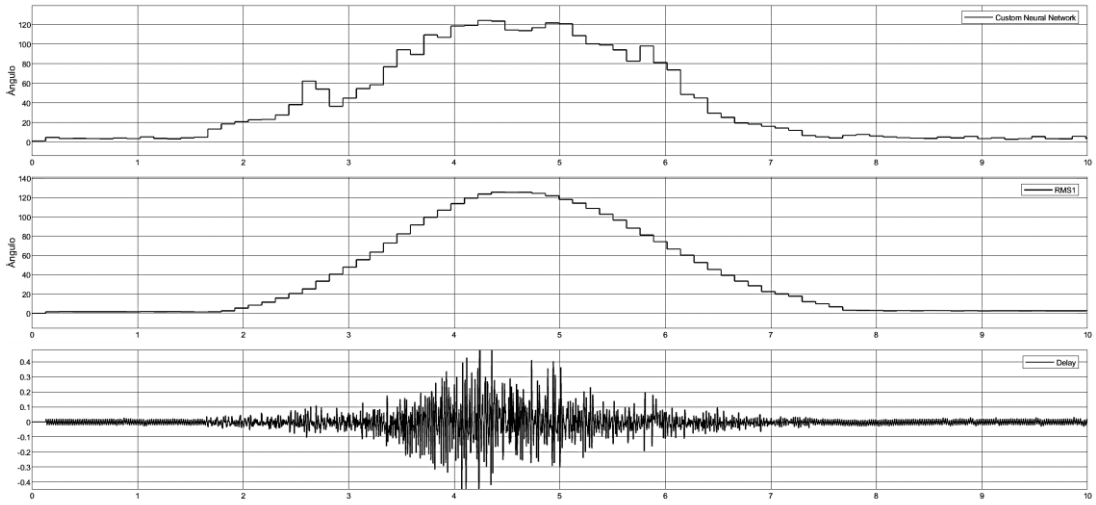



**Figura 26.** Grafica de simulación de señal de impulso 2.

En la ejecución de la simulación de la señal de impulso 2, se puede observar y analizar el comportamiento diferente de la red neuronal artificial a comparación de la señal de impulso 1. Se denota la diferencia del ángulo deseado y su respuesta ante esta, dando un comportamiento distinto e ideal para en la respuesta de control.

Esto permite conocer mejor manera la configuración de la red neuronal artificial, dando conocimiento a su estado adaptable a diferentes señales de impulso. Posteriormente se analizará el comportamiento de una última señal de impulso en la Figura 27, para estar al tanto de su adaptabilidad a una señal con un ángulo máximo deseado mayor a las señales de impulso anteriormente analizadas.

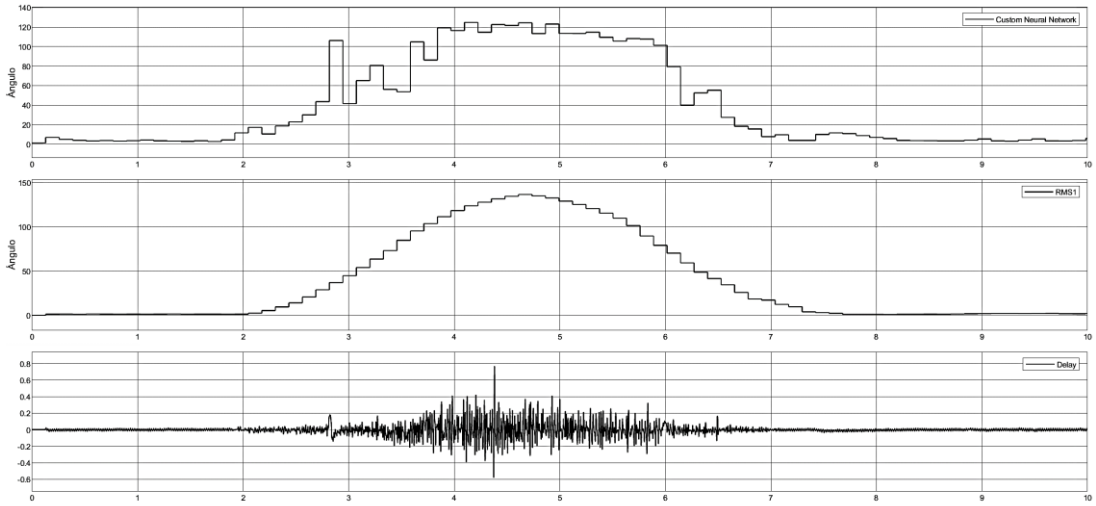



**Figura 27.** Grafica de simulación de señal de impulso 3.

Tomando estos resultados como pauta para el comportamiento de la red neuronal artificial y notando su producción controlada del ángulo deseado, se realiza la ejecución del sistema embebido en los diferentes tipos de placas de desarrollo.

Para dicha ejecución se toma las señales de impulso previamente guardados en la base de trabajo de Matlab. Con fin de observar y analizar el comportamiento del sistema de control en cada una de las placas de desarrollo, para realizar la comparación con cada una de ella y determinar la mejor respuesta existente.

Esto con base en la eficiencia al momento de ejecutar el procedimiento de control de la señal de impulso para un exoesqueleto de aumento de fuerza del miembro inferior. Para lo cual se debe cotejar diferentes aspectos de cada respuesta obtenida, los cuales son: el tiempo de muestreo de la placa de desarrollo y el comportamiento de la red neuronal artificial con respecto al ángulo resultante posterior al procesamiento de la señal.

Dicha ejecución se realiza con la señal de impulso 1 en las diferentes placas de desarrollo. En primer lugar, la placa Arduino Mega 2560, la cual es la placa de desarrollo más comúnmente utilizada a nivel de proyectos Mecatrónicos tanto por su facilidad de uso y desarrollo, como su disponibilidad en el mercado.

La primera verificación por ejecutar, se la realiza con la señal de impulso 1 con un tiempo de muestreo de 1 milisegundo, el cual permite el máximo número de lecturas que logra realizar la placa Arduino Mega 2560 por la configuración de su hardware.

Esto se lo realiza con un tiempo configurado a 10 segundo, lo cual permite observar tanto la señal de impulso como todo el trayecto de la red neuronal simulada en el entorno, así como la respuesta obtenida posteriormente al procesamiento de la placa de desarrollo.

Como se puede observar en la Figura 28, a simple vista el resultado obtenido al realizar el procesamiento de la señal de impulso 1 en la placa de desarrollo es idéntico al ángulo obtenido en la simulación del entorno. Esto da la explicación de un comportamiento ideal para el sistema de control del exoesqueleto en uso cotidiano.

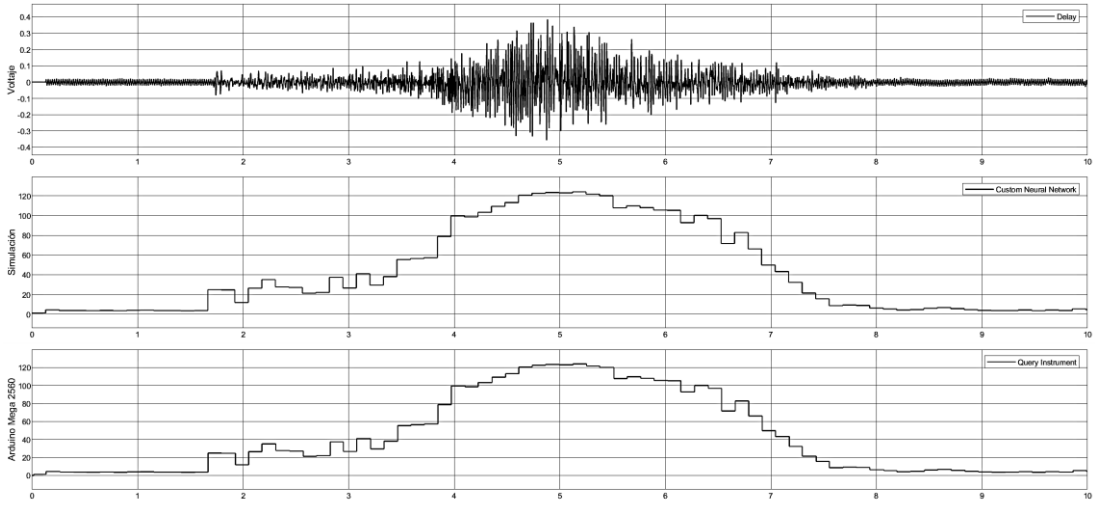



**Figura 28.** Grafica de comportamiento de la red neuronal artificial en la placa de desarrollo Arduino Mega 2560 ante la señal de impulso 1.

Al realizar un análisis minucioso de los valores obtenidos de la placa Arduino Mega 2560, se denota una variación en el ángulo con respecto al tiempo. Esto a razón del tiempo de respuesta de la señal por parte de la placa de desarrollo, puesto que los valores obtenidos poseen un retraso a comparación de la simulación del entorno.

Este retraso se manifiesta en la Figura 29, en la cual se observa un retardo en la etapa de inicialización del sistema de control, en tal sentido la gráfica obtenida por parte del procesamiento de la señal en la placa Arduino Mega 2560 da principio con un valor neto nulo a razón de la inexistencia de señal de impulso en esta etapa.

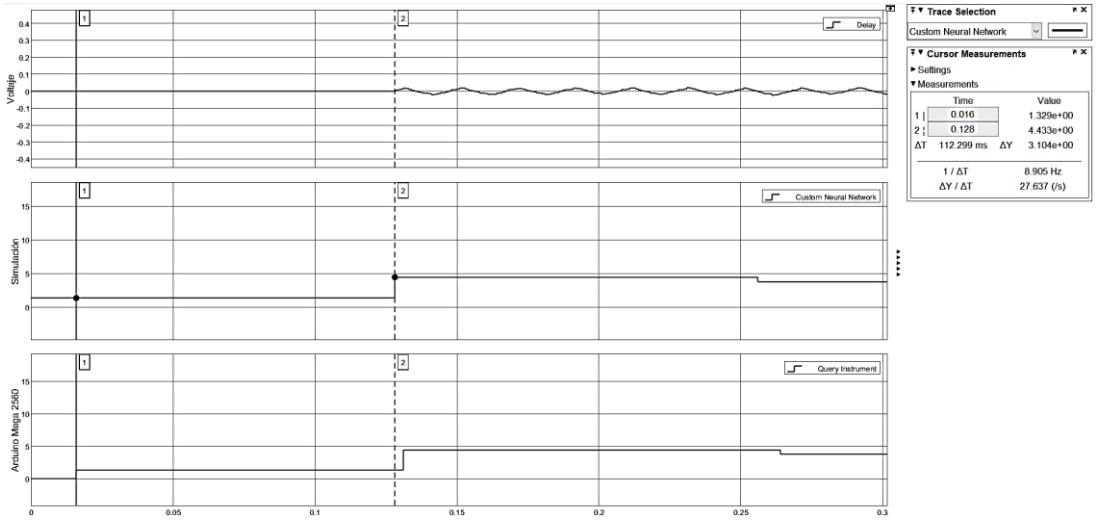



**Figura 29.** Grafica de inicialización con retardo de valor nulo de la placa de desarrollo Arduino Mega 2560 ante la señal de impulso 1.

Al seguir dicha gráfica, se observa la primera variación del ángulo nulo luego de los primeros 24 milisegundos de la inicialización de ejecución del sistema de control en la placa de desarrollo. Esta variación, como se visualizar en la Figura 30, posee una diferencia mínima de 0.1 milisegundos en el ángulo obtenido de la placa de desarrollo con respecto a la simulación del entorno en la primera variación existente, mientras que en la segunda variación de ángulo por parte del hardware es de 0.2 milisegundos a motivo de la estructura y procesador de la placa Arduino Mega 2560.

Posteriormente al realizar todas estas observaciones, se indica un correcto funcionamiento del sistema de control implementado en la placa de desarrollo Arduino Mega 2560, por motivo de poseer un retardo mínimo de 500 milisegundos deseados en el sistema de control con respecto al tiempo y una variación despreciable del ángulo en cada cabio de este.

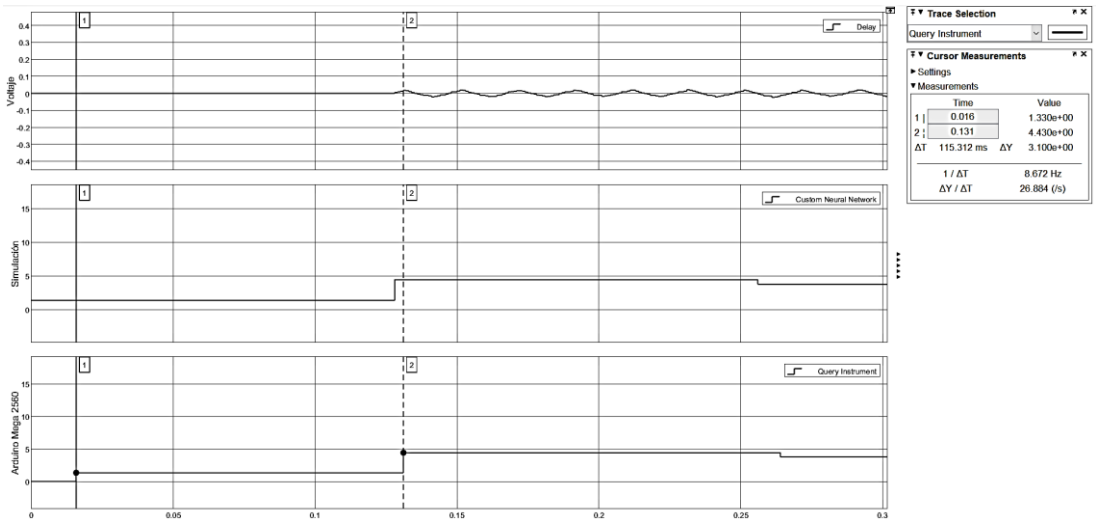



**Figura 30.** Grafica de retardo de la placa de desarrollo Arduino Mega 2560 ante la señal de impulso 1.

A continuación, se realizó el mismo procedimiento de ejecución del sistema de control de la señal de impulso 1 con la diferencia de la placa de desarrollo la cual será la placa Arduino Due. Con el fin de analizar el comportamiento del sistema de control con diversas placas de desarrollo.

En la Figura 31, se observa una reacción idéntica de la placa Arduino Due a comparación de la reacción generada por la placa Arduino Mega 2560. La respuesta obtenida de la placa Arduino Due ante la señal de impulso 1 es similar a los valores resultantes de la simulación del entorno, demostrando un comportamiento ideal para el procesamiento de señal y control.

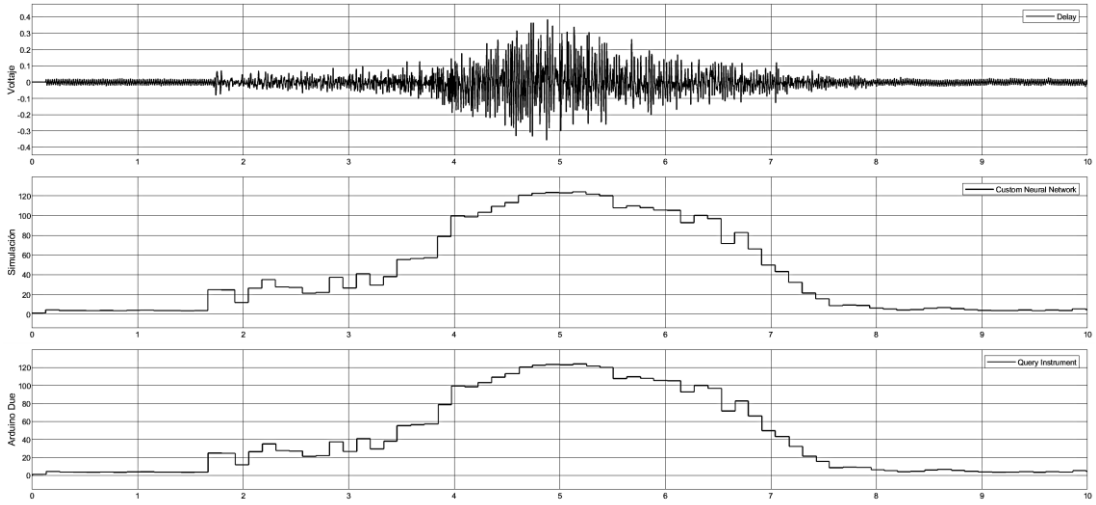



**Figura 31.** Grafica de comportamiento de la red neuronal artificial en la placa de desarrollo Arduino Due ante la señal de impulso 1.

A simple vista se logra observar un mínimo retardo de la señal de la placa de desarrollo, por motivo del muestreo realizado. Por tal motivo, se realiza un análisis minucioso con respecto al tiempo para determinar el desplazamiento de la reacción del sistema de control en el hardware y conocer el periodo existente con respecto a un entorno simulado.

De acuerdo con la Figura 32 y Figura 33, se logra determinar de manera más detallada el retardo existente en la placa de desarrollo y comprender el tiempo aproximado de reacción de esta. Esto tomando en cuenta la alteración existente en cada cambio de dato por parte de la señal de impulso 2, y con el tiempo de muestreo configurado para la obtención de la mayor velocidad de respuesta posible.

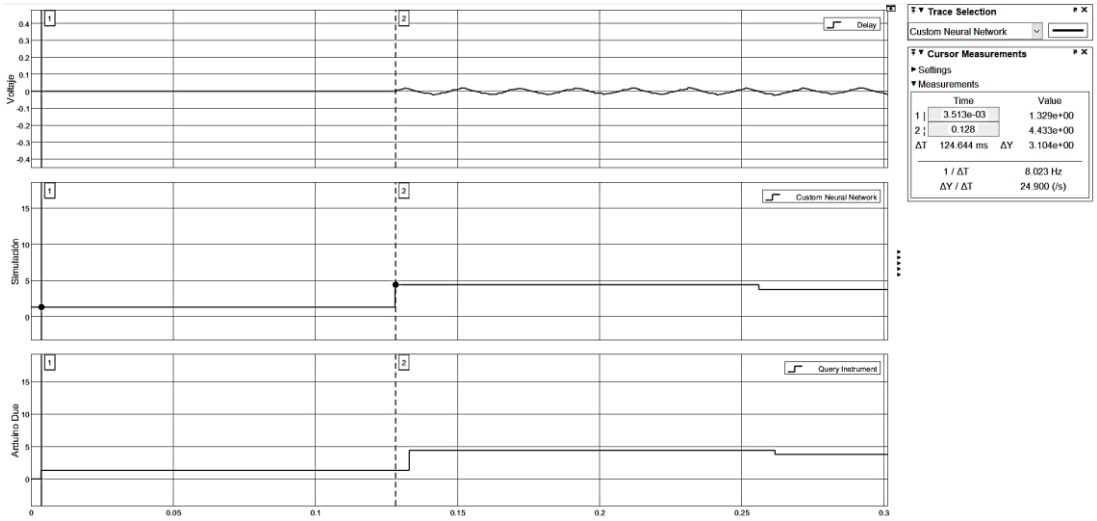



**Figura 32.** Grafica de inicialización con retardo de valor nulo de la placa de desarrollo Arduino Due ante la señal de impulso 1.

A diferencia de la Figura 33, donde se puede observar el retardo existente de la placa Arduino Due por motivo del tiempo de muestreo que realiza. Este retardo existente en la respuesta del ángulo obtenido es mínimo, lo cual determina un procesamiento ideal con respecto a la red neuronal artificial.

El tiempo de retraso en responder con un valor de salida del ángulo por parte de la placa Arduino Due es de 65 milisegundo, demostrando un mayor tiempo reacción ante la señal de impulso a comparación de la placa Arduino Mega 2560.

Cabe recalcar que este tiempo de respuesta por parte del Arduino Due tiene un comportamiento ideal, a razón de colocarse por debajo de los 500 milisegundos de reacción deseados. Esto demuestra poseer un funcionamiento deseado para el exoesqueleto.

Con respecto al valor obtenido por parte de la placa de desarrollo de respuesta del ángulo, este posee una mínima variación en la primera reacción con respecto al ángulo obtenido en el entorno simulado. Dicho valor es de 1 segundo, lo cual demuestra un procesamiento ideal de la señal

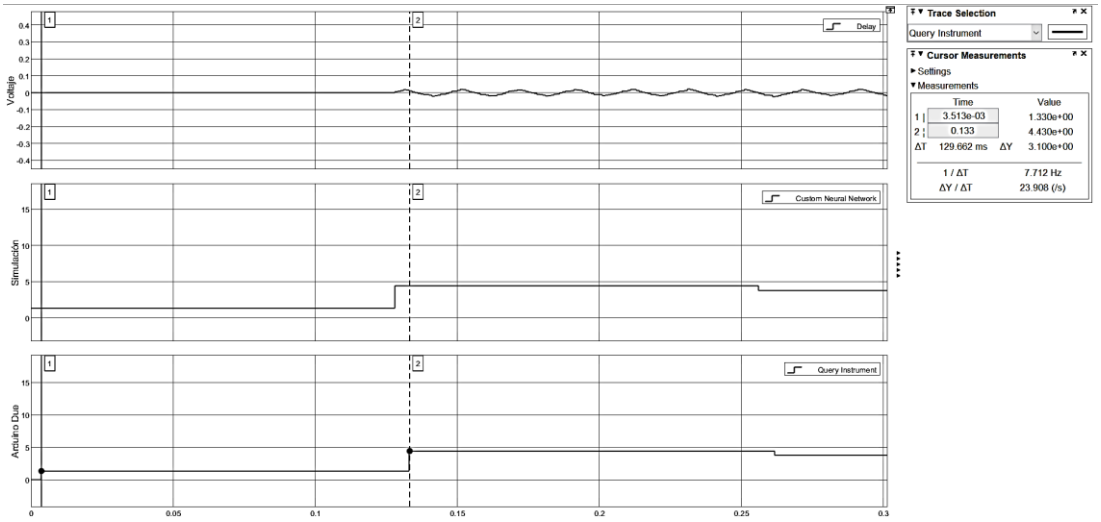



**Figura 33.** Grafica de retardo de la placa de desarrollo Arduino Due ante la señal de impulso 1.

Posteriormente, se logra observar un retraso mínimo en el valor del tiempo y ángulo con respecto a la segunda reacción del valor obtenido de la simulación del entorno. Esta variación de tiempo es insignificante al tener un valor de 7 milisegundos y una variación del ángulo de 3 segundo.

Esta observación determina que la placa Arduino Due en el transcurso del tiempo tiene una mejor reacción, a razón de lograr procesar valores de manera más eficiente a comparación de la placa Arduino Mega 2560.

Por último, en esta etapa, se realiza el mismo procedimiento de observación y análisis de la respuesta del sistema de control ante la señal de impulso 1, con la diferencia de la placa de desarrollo y el método PIL utilizados. Dicha placa de desarrollo es la Rapsberry Pi 3 B+, la cual es la más cara en el mercado.

Para ejecutar el sistema de control por medio de PIL en la placa Rapsberry Pi 3 B+, se utiliza el método directo para embeber dicho sistema en la placa de desarrollo. Posteriormente de haber realizado este procedimiento, se ejecuta todo el sistema de control dentro de la placa Rapsberry Pi 3 B+ a la par de la gráfica del ángulo deseado.

Esto permite determinar el comportamiento de la placa de desarrollo junto con la red neuronal artificial, y determinar la eficiencia de su funcionamiento con la arquitectura y procesador que posee.

Como se logra observar en la Figura 34, al haber realizado un proceso PIL directo y poseer la posibilidad de importar los valores de la señal de impulso 1, este no posee ningún retardo por motivo de la inexistencia de tiempo de muestreo.

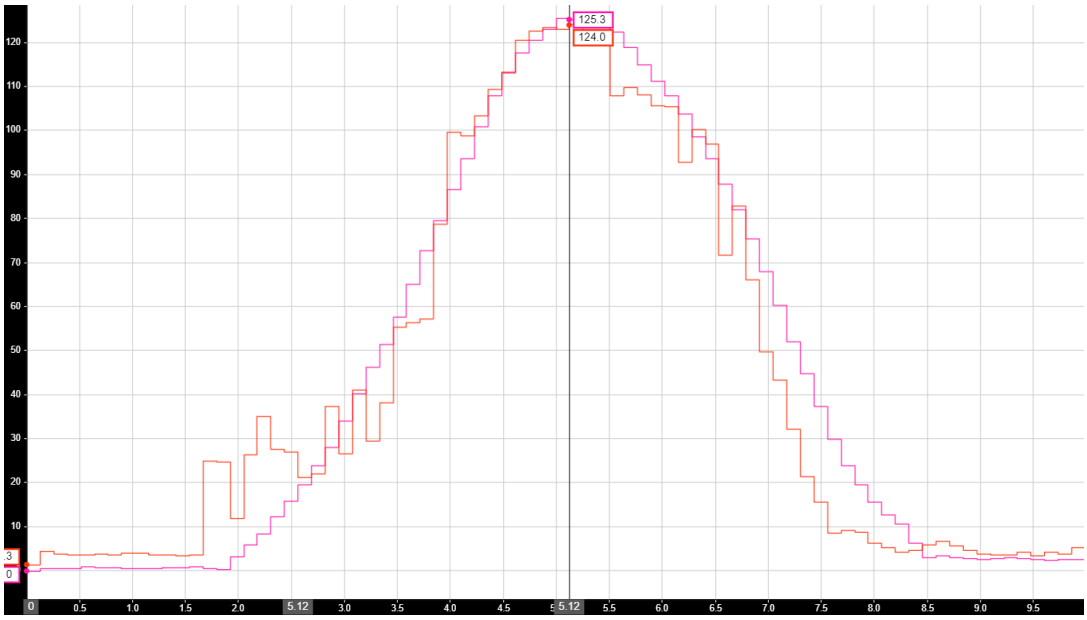



**Figura 34.** Grafica de comportamiento de la red neuronal artificial en la placa de desarrollo Raspberry Pi 3 B+ ante la señal de impulso 1.

Esto demuestra que dicha placa posee un retardo de reacción por motivo de muestreo en campo, pero sin logra indicar un comportamiento más ideal con respecto a las anteriores placas de desarrollo. Por tal motivo no se puede realizar un análisis con respecto al tiempo de esta placa, pero cabe recalcar que su respuesta ante la señal de impulso posee mayor velocidad por motivo del procesador que posee.

Como se observar, la red neuronal artificial posee un comportamiento deseado ante la señal de impulso 1 en la placa Raspberry Pi 3 B+, esto por motivo de que dicha placa de desarrollo por su configuración posee la habilidad de un comportamiento equiparable a un ordenador de escritorio.

Esta respuesta obtenida indica una reacción favorable para la señal de impulso 1, de modo que no generará ninguna variación excesiva del ángulo, ni tendrá un retraso notable en la respuesta del sistema de control dando una fidelidad ante la señal.

Posteriormente de haber realizado el análisis del comportamiento de la red neuronal ante la señal de impulso 1 en las diferentes placas de desarrollo, se realiza otro procedimiento de análisis para una señal de impulso diferente, lo cual dará un mayor conocimiento ante diversas señales existentes.

El segundo procedimiento de análisis se realiza con la señal de impulso 2 guardado anteriormente en la base de trabajo de Matlab. Esta señal de impulso posee una diferencia notable ante la señal de impulso 1, lo cual permite observar el comportamiento de la red neuronal ante una señal de impulso con distinto tiempo de muestreo.

En primer lugar, la placa Arduino Mega 2560, se ejecuta con el mismo procedimiento llevado a cabo anteriormente con la señal de impulso 1. Al momento de efectuar el procesamiento de la señal de impulso 2 por la red neuronal artificial, se toma los datos para la gráfica demostrativa de su respuesta.

Como se puede observar en la Figura 35, la existencia de tiempo de muestreo sigue afectando en el resultado obtenido del ángulo por parte de la placa Arduino Mega 2560. Aunque dicha afectación no produce ningún problema al momento de realizar el procesamiento de la señal, si demuestra la eficiencia de la placa de desarrollo para la toma y desarrollo de valores de entrada.

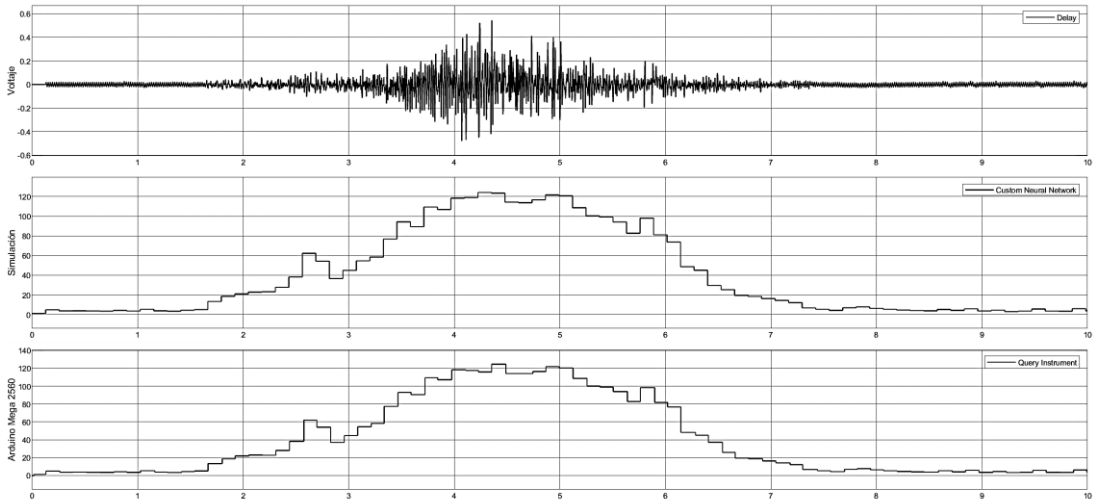



**Figura 35.** Grafica de comportamiento de la red neuronal artificial en la placa de desarrollo Arduino Mega 2560 ante la señal de impulso 2.

En la gráfica obtenida resultante del procesamiento de la señal de impulso 2 por la placa de desarrollo, se denota un retardo mínimo el cual después de un análisis minucioso de los valores adquiridos se aproxima un valor de 8.4 milisegundo en la respuesta por parte del hardware ante una variación de datos de entrada.

Dicha variación es por parte del ángulo, el cual es el resultante del procesamiento de señal de impulso 2. Esta variación se denota por la inicialización con valor nulo por la inexistencia de datos al segundo 0 y se logra observar de manera más detallada en la Figura 36.

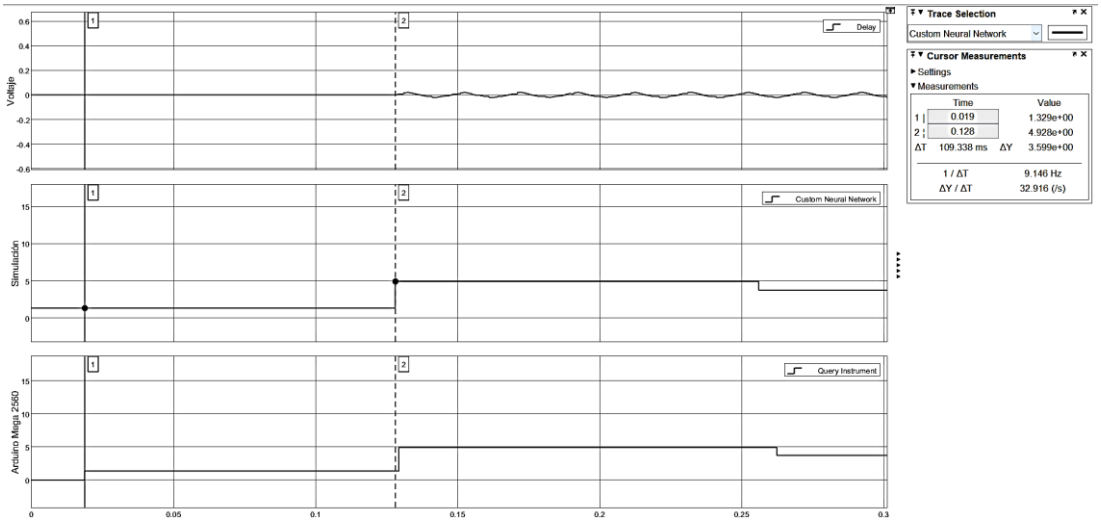



**Figura 36.** Grafica de inicialización con retardo de valor nulo de la placa de desarrollo Arduino Mega 2560 ante la señal de impulso 2.

Como se observa, al poseer una alteración más veloz en la etapa de inicio de la señal de impulso a comparación de la señal de impulso 1, permite disminuir el retardo existente por motivos del muestreo a un valor insignificante. Esto genera una reacción en cadena por parte de procesamiento, obteniendo valores de ángulo por la placa de desarrollo muy similares a la simulación del entorno con respecto al tiempo.

Todo esto da a conocer un procesamiento ideal por parte de la placa Arduino Mega 2560, esto queriendo decir un funcionamiento adecuado por parte del sistema de control para el exoesqueleto.

Como se puede observar en la Figura 37, la variación existente entre los valores de ángulo de la simulación del entorno y los valores obtenidos posteriormente al procesamiento de la señal de impulso 2 en el hardware, son aproximadamente similares a los datos obtenidos con el procedimiento por parte de la señal de impulso 1 anteriormente analizada.

Dichos valores de variación son de 1 segundo en la primera alteración por parte de la placa de desarrollo y de 2 segundos en su segunda alteración. Esto demuestra que la variación por parte de la señal de impulso 1 y señal de impulso 2 no genera una respuesta diferente con respecto al porcentaje deseado de la alteración del ángulo, si no explica el comportamiento de la red neuronal y el tiempo de muestreo de la placa de desarrollo ante una señal con cambios lentos en la etapa inicial.

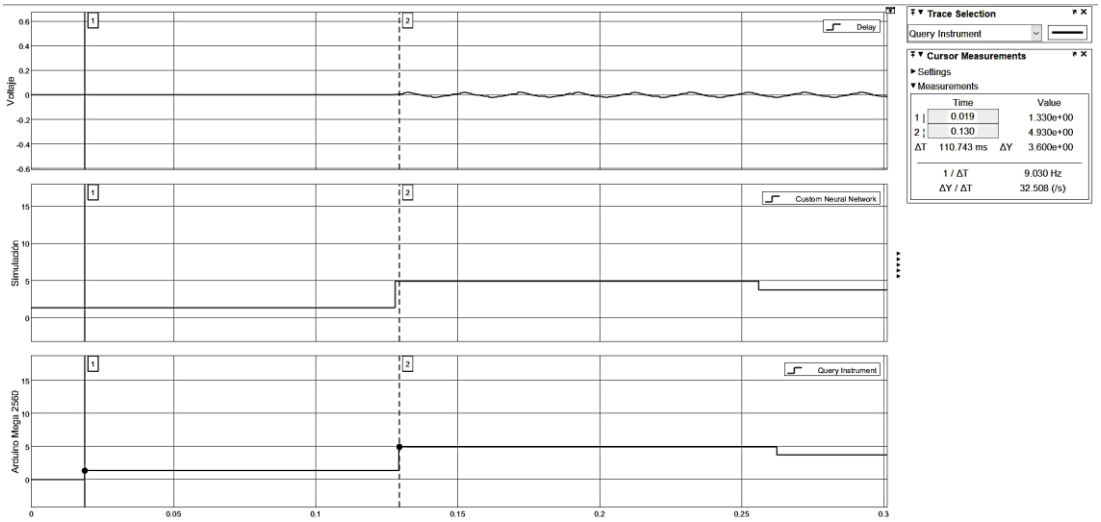



**Figura 37.** Grafica de retardo de la placa de desarrollo Arduino Mega 2560 ante la señal de impulso 2.

Siguiendo la comprobación del comportamiento de la red neuronal en las diferentes placas de desarrollo, y conocer la variación posible para cada una de ellas ante diferentes señales de impulso. Se realiza el mismo procedimiento para la observación y análisis del comportamiento del sistema de control en la placa Arduino Due.

En esta etapa, la configuración por parte de la placa de desarrollo para el procesamiento de la señal de impulso 2 es idéntica a la configuración previamente realizada para el análisis de la señal de impulso 1. De este modo, se logrará observar, analizar y compara el comportamiento de la red neuronal artificial en la placa Arduino Due con una variación en la señal de entrada, la cual es más veloz a comparación de la señal de impulso 1.

En la Figura 38, se puede observar que los valores obtenidos por parte de la placa de desarrollo y los valores generados en la simulación del entorno son idénticos a simple visto, esto demuestra que la variación de dichos valores con respecto al tiempo es insignificante por poseer una aproximación a la igualdad entre ellos.

Por consiguiente, se puede deducir que el tiempo de retardo del ángulo y la variación del valor de este obtenido posterior al procesamiento de la señal de impulso 2 en la placa Arduino Due es mínimo. Esto se observa en un análisis meticuloso de la gráfica generada, para visualizar de mejor manera los valores generados por parte de la placa de desarrollo y la simulación del entorno del sistema de control.

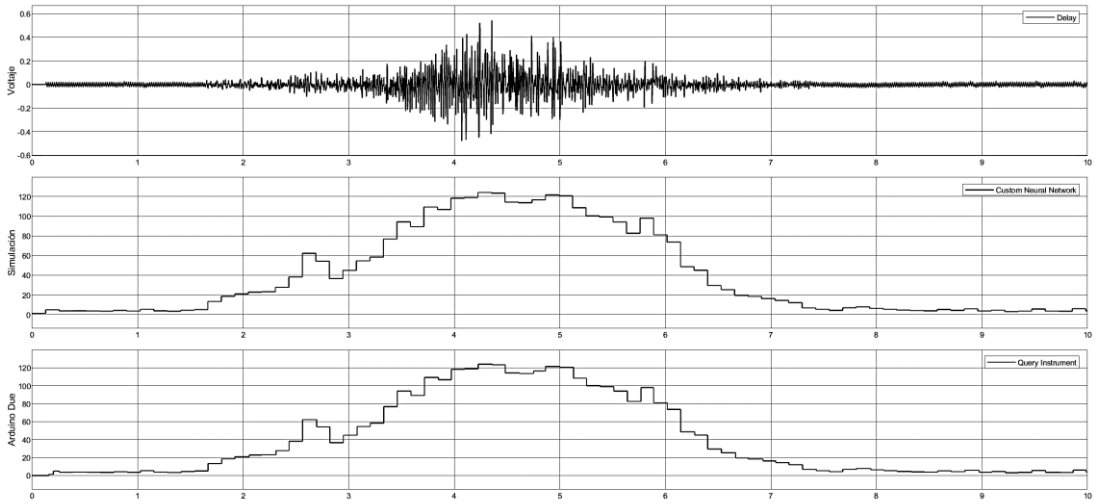



**Figura 38.** Grafica de comportamiento de la red neuronal artificial en la placa de desarrollo Arduino Due ante la señal de impulso 2.

Dicho análisis se logra observar en la Figura 39, en la cual se denota un valor de variación del ángulo obtenido subsiguiente al procesamiento de la señal de impulso 2. Este valor se logra aproximar con la Figura 40 por motivos de la señalización utilizada para remarcar los valores generados en el transcurso del tiempo

Este valor aproximado de 33 milisegundo en la primera variación de la señal de impulso expresa un retardo despreciable mayor a comparación del valor del ángulo obtenido con la placa Arduino Mega 2560, pero analizando los valores obtenidos de la placa Arduino Due en el transcurso del tiempo, se logra observar una segunda variación del ángulo por parte del procesamiento del hardware.

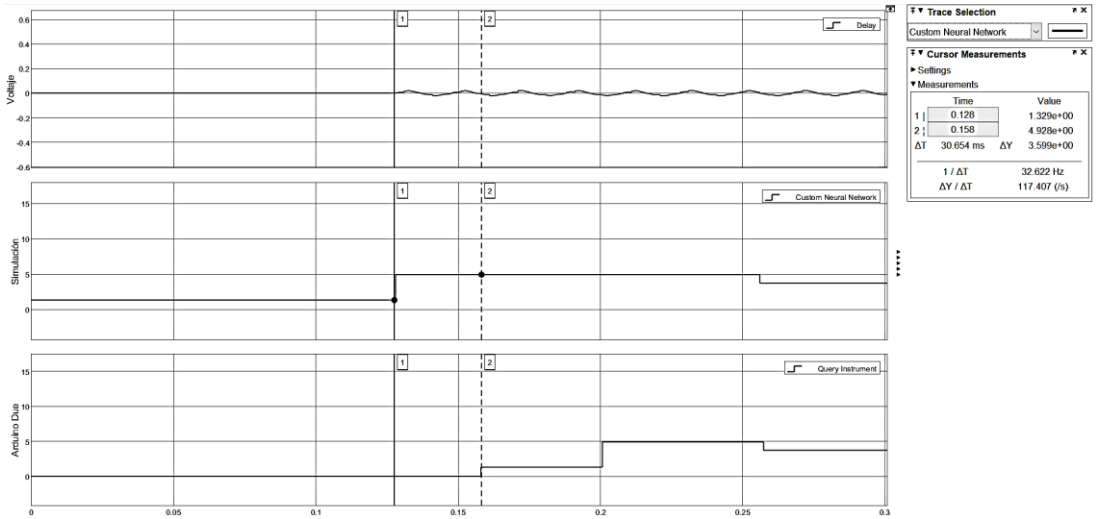



**Figura 39.** Grafica de inicialización con retardo de valor nulo de la placa de desarrollo Arduino Due ante la señal de impulso 2.

Dicha variación al ser aproximada se observa un retardo del ángulo, con respecto a los valores obtenidos de la simulación del entorno, mucho menor a la primera variación analizada. Este valor, posterior al ser analizado se logra observar un valor similar a la segunda variación del ángulo en la placa Arduino Mega 2560, el cual es de 3 milisegundos aproximadamente.

Esto logra expresar que la placa de desarrollo Arduino Due consigue una estabilidad en relación con el tiempo, lo cual adquiere una menor alteración del ángulo obtenido en el procesamiento de la señal de impulso a lo largo del tiempo de ejecución.

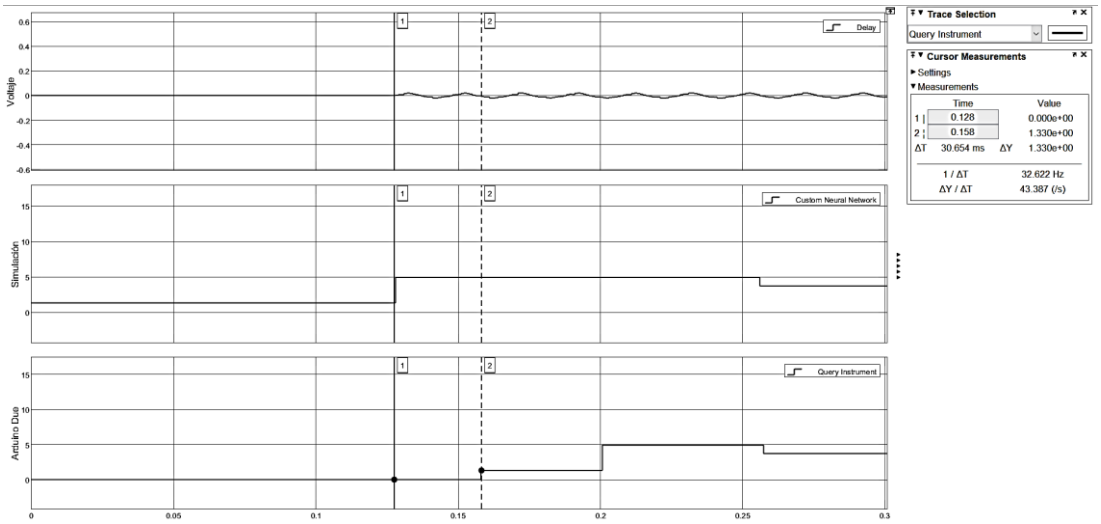



**Figura 40.** Grafica de retardo de la placa de desarrollo Arduino Due ante la señal de impulso 2.

Por otro lado, la última observación y análisis a realizar con la señal de impulso 2, se la debe llevar a cabo con la placa de desarrollo Raspberry Pi 3 B+. Esta debe poseer la misma configuración la señal de impulso 1 para el análisis y comparación de respuesta obtenidas de los diferentes procesamientos del hardware, embebida con las dos diferentes señales de impulso.

El procedimiento por realizar en la placa Arduino Due es idéntico al procedimiento que se realizó en la placa Arduino Mega 2560. Dicha ejecución se la realiza con el método de PIL directo, permitiendo exportar todos los datos de la señal de impulso 2 previamente guardado.

Esto, facilitando el procesamiento de la señal de impulso con la red neuronal artificial, otorgando la posibilidad de eliminar el tiempo de muestreo, lo cual genera un punto a favor y uno en contra ante el análisis del procesamiento.

Como se logra observar en la Figura 41, se denotan los dos puntos generados por PIL directo mencionados anteriormente.

Dichos puntos son, la ausencia de retardo por la inexistencia de tiempo de muestreo a causa del embebido directo de todo el sistema de control, así como los datos de la señal de impulso 2, y el segundo punto, el cual es tener un mejor control de la respuesta de la red neuronal artificial ante una señal de impulso con una inicialización de datos veloz.

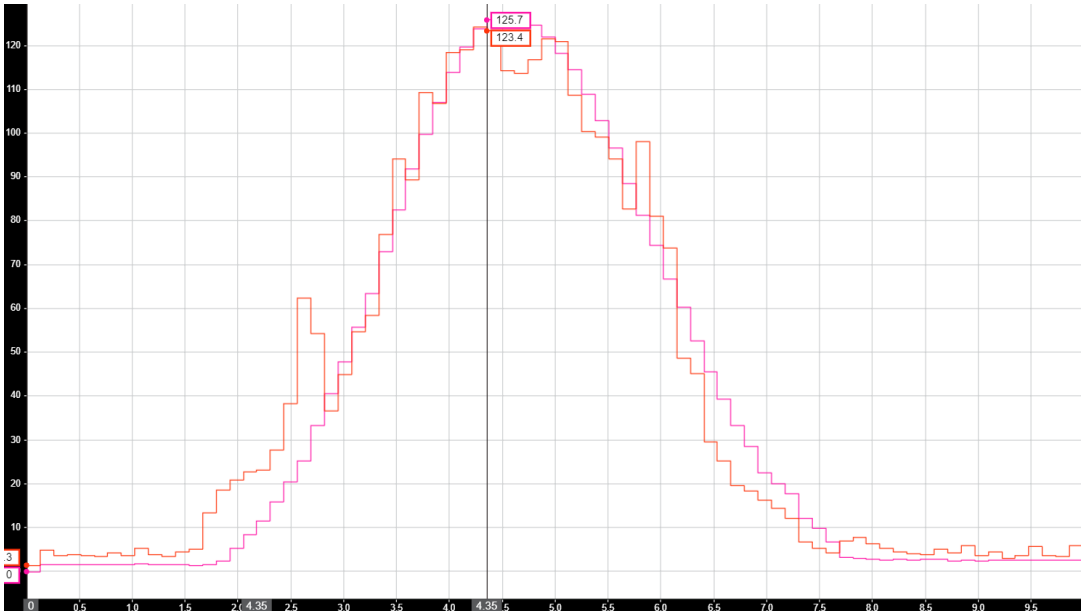



**Figura 41.** Grafica de comportamiento de la red neuronal artificial en la placa de desarrollo Raspberry Pi 3 B+ ante la señal de impulso 2.

El primero punto generado afecta de forma directa al análisis del procesamiento de la señal de impulso en la red neuronal a lo largo del tiempo de ejecución dentro de la placa de desarrollo. Esto causa una falta de variación del impulso en la etapa de inicio, lo cual no permite observar la velocidad de procesamiento ante una señal muestreada de tiempo real.

Por tal motivo, en dicha etapa no se logra observar la velocidad de cambio del ángulo generado por el procesamiento de la red neuronal, pero logra originar la posibilidad de análisis en el transcurso del tiempo para determinar el comportamiento del sistema de control, el cual se observa un seguimiento ideal al ángulo deseado. Esto demuestra un procesamiento similar a un procesador de ordenador, denotando una respuesta fluida del sistema de control, así como el poder obtener el ángulo deseado de forma controlada sin ninguna variación excedente en el transcurso del tiempo.

Para concluir en esta etapa de análisis del comportamiento de la red neuronal, se realiza el mismo procedimiento en cada una de las placas de desarrollo con la diferencia de la señal de impulso. Esto para conocer el comportamiento del sistema de control ante una señal intermedia, demostrando el comportamiento deseado para el control del exoesqueleto.

Para comenzar, se realiza este proceso de observación y análisis con la señal de impulso 3 en la placa Arduino Mega 2560. Esto permitiendo conocer de mejor manera la respuesta de cada placa de desarrollo ante una última señal de impulso, generando una comprobación de cada hardware y determinando el mejor comportamiento de cada una de estas.

En la Figura 42, se logra observar un comportamiento de la red neuronal artificial similar a las señales de impulso previamente analizadas en dicha placa de desarrollo. Esta muestra un comportamiento ideal con retardo mínimo observable en la gráfica, lo cual permite conocer un funcionamiento similar a la simulación del entorno.

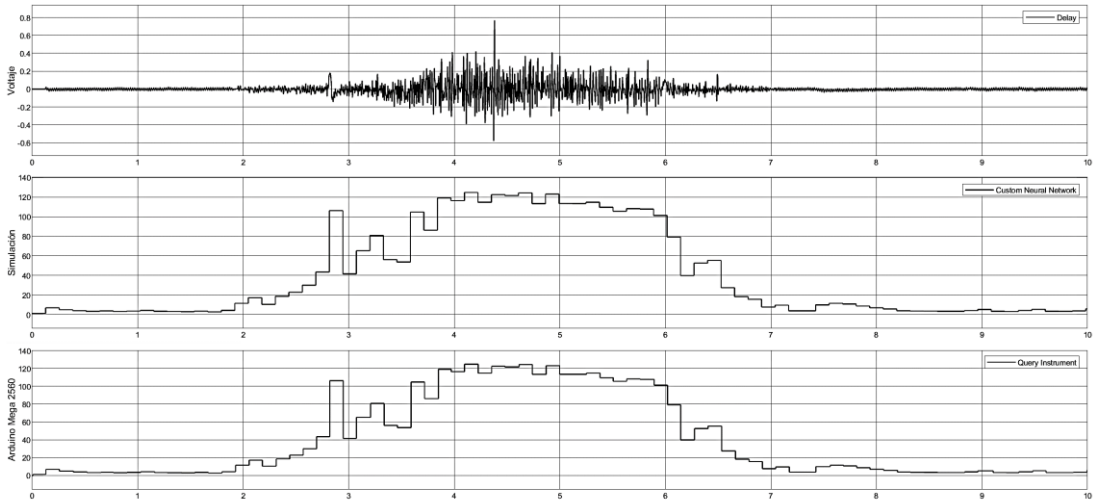



**Figura 42.** Grafica de comportamiento de la red neuronal artificial en la placa de desarrollo Arduino Mega 2560 ante la señal de impulso 3.

Como se observa, la variación generada por el tiempo de muestreo es imperceptible ante una inspección rápida por el ojo humano, de tal manera que se genera la necesidad de una inspección más minuciosa para observar la variación existente por parte de la placa de desarrollo Arduino Mega 2560.

En la Figura 43, se logra alcanzar a observar la variación del ángulo resultante por el procesamiento de la señal de impulso en la placa de desarrollo. Esta variación cambia en el transcurso del tiempo de ejecución, donde la primera es mayor a la analizada con la señal de impulso 2, pero es menor a la analizada con la señal de impulso 1, obteniendo un valor aproximado de 11 milisegundos. Al contrario, con la segunda variación del impulso, la cual posee un valor aproximado de 5 milisegundos.

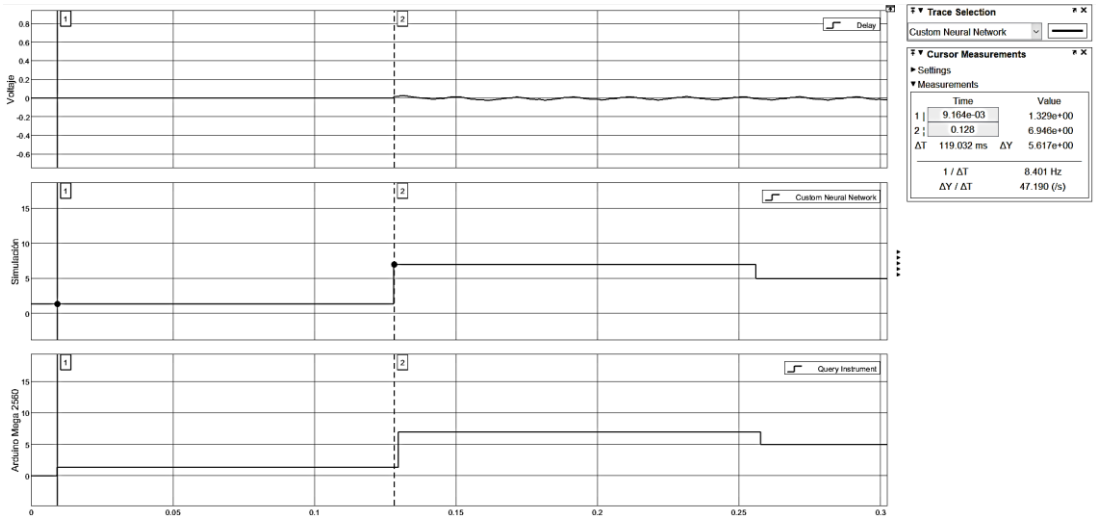



**Figura 43.** Grafica de inicialización con retardo de valor nulo de la placa de desarrollo Arduino Mega 2560 ante la señal de impulso 3.

Estos valores se logran observar en la Figura 44, obteniendo los datos resultantes del procesamiento de la señal de impulso en el hardware. Con esto se señala un comportamiento adecuado de la red neuronal artificial embebida en la placa Arduino Mega 2560, a razón de poseer información del tiempo de procesamiento y obtención de datos.

Esto denota un óptimo funcionamiento del sistema de control neuronal embebido por PIL en la placa de desarrollo Arduino Mega 2560, dando una variedad de conclusiones con respecto al funcionamiento y comportamiento de la estructura de la placa de desarrollo, en especial del procesamiento realizado por el procesador ATmega2560.

Por consiguiente, se logra obtener diferentes razones para el uso de la placa Arduino Mega 2560, el cual permite un sistema de control de exoesqueleto por medio de PIL ante diversas variaciones posibles generadas por diversos tipos de impulsos generados por señales electromiográficas.

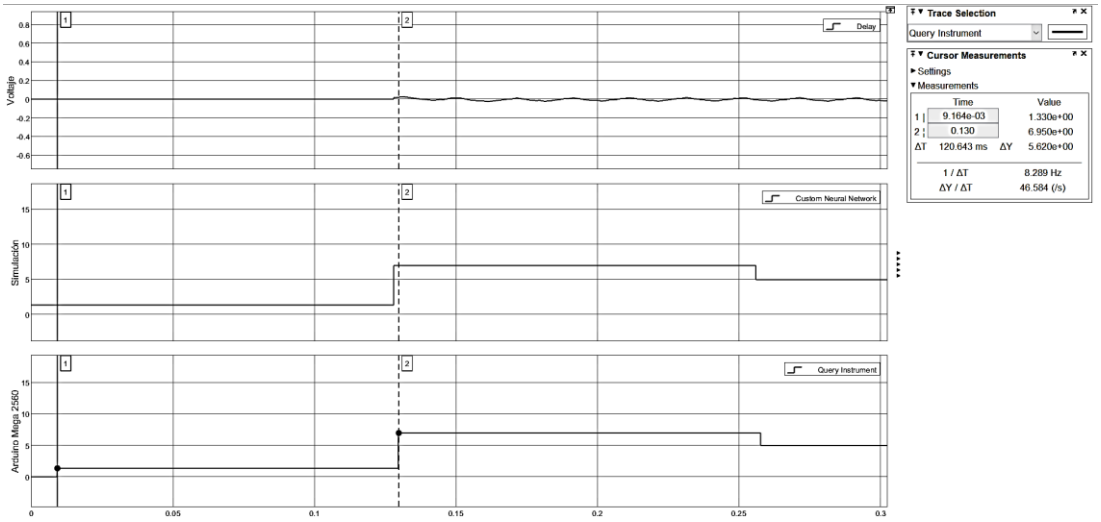



**Figura 44.** Grafica de retardo de la placa de desarrollo Arduino Mega 2560 ante la señal de impulso 3.

Siguiendo el proceso de análisis de la señal de impulso 3, se lleva a cabo el procesamiento de dicha señal en la placa de desarrollo Arduino Due. Esto permitiendo observar todas las variaciones posibles ante señales de diferente índole y el comportamiento del sistema de control embebido por PIL en dicha placa.

Al realizar el procedimiento respectivo, se obtiene la gráfica que representa los valores obtenidos posterior al procesamiento de la señal de impulso 3 con el retardo del resultado extraído de la placa de desarrollo por motivo del tiempo de muestreo.

Todos estos datos se logran observar en la Figura 45, realizando un énfasis en el retardo del ángulo resultante por parte de la placa, el cual ante el ojo humano es inexistente por motivo de un excelente procesamiento y tiempo de muestreo realizado. Esto generar la necesidad de un análisis en el tiempo a una escala mínima de milisegundos, puesto que no existe la posibilidad de realizar una observación adecuada a una escala de segundos.

Esto demuestra un sistema de control optimo por parte de la placa de desarrollo Arduino Due, y un PIL eficiente, a razón de poseer una respuesta próxima a la igualdad con respecto a la simulación del entorno, el cual no posee tiempo de muestreo por motivo de disponer los datos de la señal de impulso previamente guardada en la base de trabajo de Matlab.

A continuación, se realiza una ampliación del análisis para lograr la observación y determinar el valor aproximado de la variación del ángulo resultante por el procesamiento del hardware, dando de este modo un valor a señal para conocer la velocidad de procesamiento de la placa de desarrollo y su estructura.

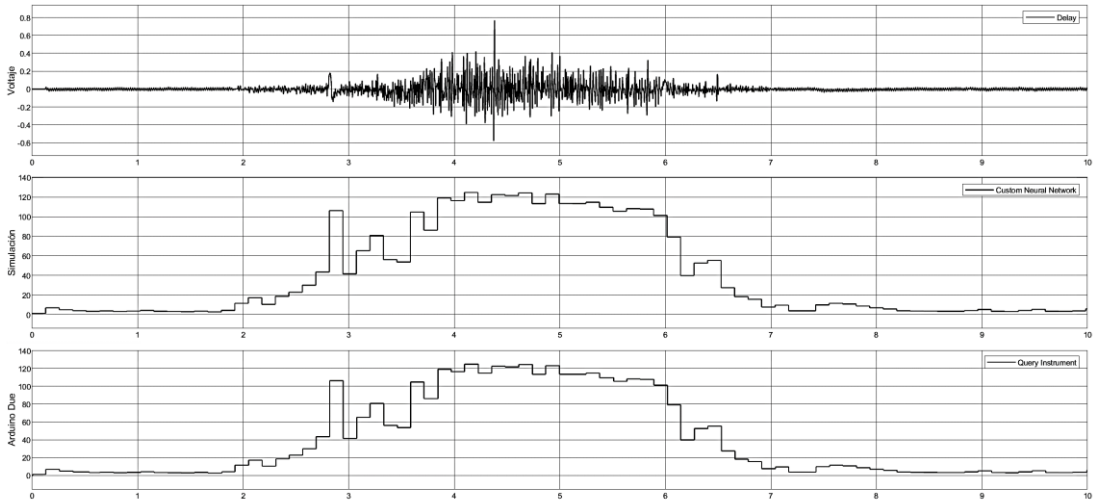



**Figura 45.** Grafica de comportamiento de la red neuronal artificial en la placa de desarrollo Arduino Due ante la señal de impulso 3.

Dicho análisis minucioso se logra observar en la Figura 46, en la cual se observa que la primera variación del ángulo obtenido como resultado del procesamiento de la señal de impulso 3 es irrisible, por motivo de poseer un valor muy mínimo.

Este valor de la variación se logra obtener por ayuda de la gráfica, la cual permite obtener datos que el ojo humano no logra percibir. Con toda esta ayuda se logra aproximar el valor resultante de la variación por motivo de tiempo de muestreo, el cual es de 3.2 milisegundo desde la etapa de inicio del procesamiento de la señal con respecto a la primera variación del ángulo generada en el hardware.

Esto denota una obtención de la señal de impulso más fluido, lo cual es causado por la estructura de la placa de desarrollo y el procesador perteneciente a dicha placa.

Posteriormente con el seguimiento en el transcurso del tiempo del ángulo resultante del procesamiento de la señal de impulso en la Figura 45, se logra una segunda variación de datos más notable a comparación de la primera. Esto genera la necesidad de conocer el valor de la variación del ángulo con respecto al tiempo, por tal motivo se analiza con la misma herramienta utilizada para la primera.

Lo cual permite estimar el valor del tiempo retardo en la gráfica, dando un mejor conocimiento del comportamiento de la placa de desarrollo con la señal de impulso 3.

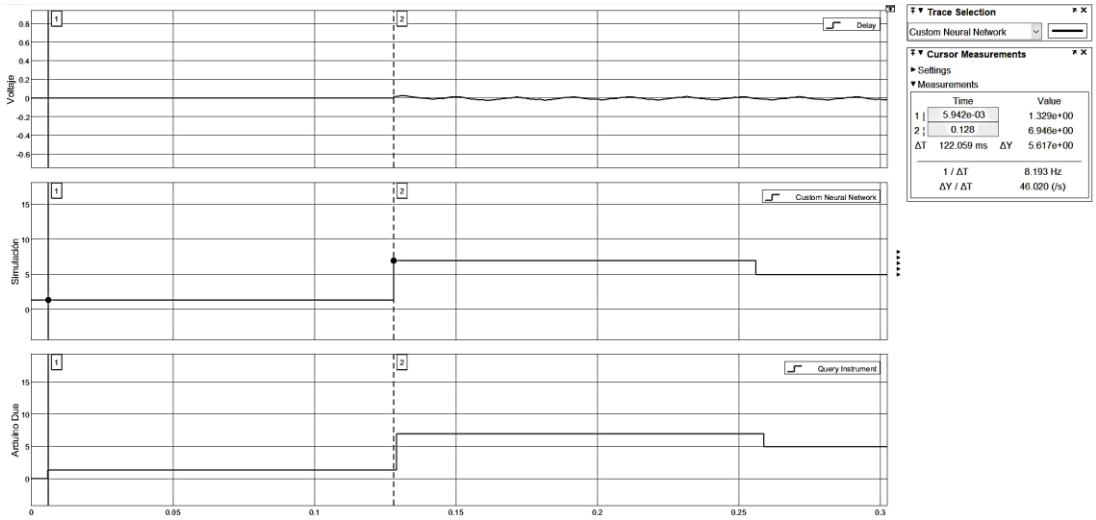



**Figura 46.** Grafica de inicialización con retardo de valor nulo de la placa de desarrollo Arduino Due ante la señal de impulso 3.

El análisis anteriormente mencionado, se observa en la Figura 47 con la herramienta pertinente para conocer el valor de la variación de la segunda alteración del ángulo con respecto al procesamiento del hardware.

Este valor aproximado es de 3 milisegundos, dando a conocer el comportamiento de la placa Arduino Due, el cual es similar al comportamiento a lo largo del tiempo a comparación de la placa de desarrollo Arduino Mega 2560. A su vez, se examina el valor resultante del ángulo obtenido como respuesta del procesamiento de la placa Arduino Due, el cual tiene un valor idéntico en el porcentaje de cambio del ángulo en relación con la placa Arduino Mega 2560.

Este valor en la primera alteración del sistema de control está definido con el valor aproximado de 36 segundos en el ángulo resultante, y con un valor semejante en la segunda alteración del sistema de control del valor obtenido con la placa de desarrollo Arduino Mega 2560, el cual está determinado con el valor aproximado de 14,4 segundos del ángulo resultante adquirido por la generación de datos por el procesamiento de la placa Arduino Due.

Toda esta información denota que la diferencia de estructura, como de procesador incorporado de la placa Arduino Due y la estructura de la placa Arduino Mega 2560, no realiza ningún cambio notable al momento de ejecutar un sistema de control para un exoesqueleto.

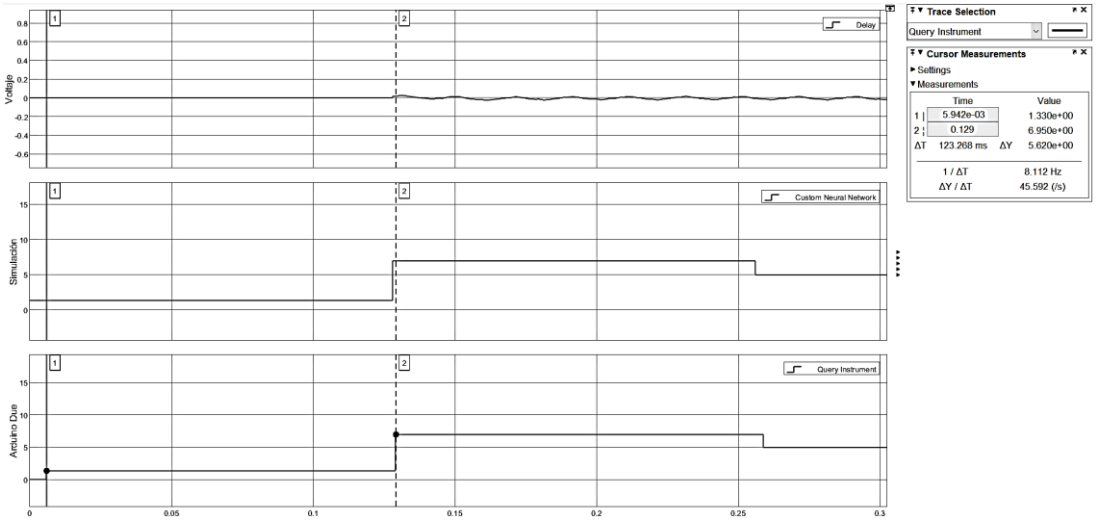



**Figura 47.** Grafica de retardo de la placa de desarrollo Arduino Due ante la señal de impulso 3.

Finalmente, en esta etapa de análisis con la señal de impulso 3, se genera el procesamiento de la señal de impulso en la placa de desarrollo Raspberry Pi 3 B+. Esta placa no se altera su sistema de control, por lo cual la configuración realizada no cambia, su única alteración es la señal embebida que sufre un cambio por la señal de impulso 3.

Por tal motivo, al tener un sistema de control embebido por PIL directo se debe importar los valores de dicha señal de impulso para obtener el comportamiento de la red neuronal artificial ante una tercera diferente señal de impulso en esta.

Posteriormente de haber realizado el aumento de datos de la señal de impulso en la placa Raspberry Pi 3 B+, se ejecuta el procesamiento de la señal para el análisis de este. En la Figura 48, se observa la variación generada del ángulo resultante por parte del sistema de control con la red neuronal artificial al momento de efectuar el procesamiento de señal en el hardware.

Se marca variación existente en el ángulo mayor del sistema de control en la placa de desarrollo a comparación del ángulo deseado, esto da la información del comportamiento de la red neuronal artificial y demuestra la inexistencia de algún posible error por sobrepasar el ángulo deseado.

Esta información denota un excelente comportamiento del sistema de control en la placa Raspberry Pi 3 B+, por el motivo de poseer una estructura mejorada a comparación de las placas de desarrollo previamente analizadas. El valor obtenido en la variación del ángulo es aproximado de 12 grados y 12 minutos, dando una confiabilidad en el sistema de control del exoesqueleto.

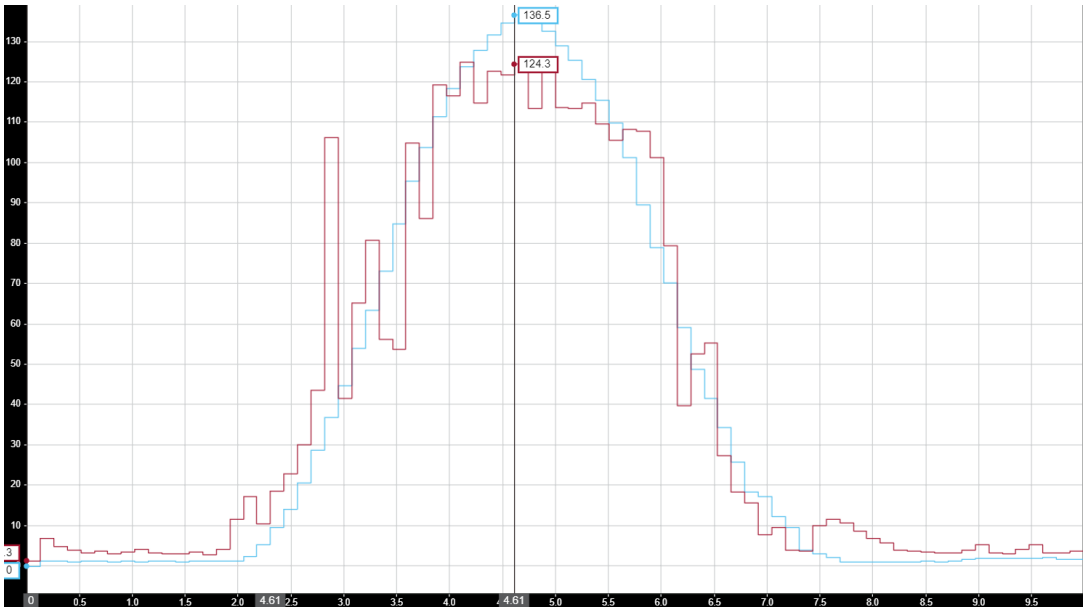



**Figura 48.** Grafica de comportamiento de la red neuronal artificial en la placa de desarrollo Raspberry Pi 3 B+ ante la señal de impulso 3.

## 3.2. COMPARACIÓN DE PROCESAMIENTO

En este punto, se realizó la comparación de los resultados obtenidos en el procesamiento de los diferentes tipos de señales de impulso en las distintas placas de desarrollo usadas en el transcurso del documento.

Esta comparación permitirá analizar de mejor manera la eficiencia existente en cada procesamiento de cada una de las placas de desarrollo con el sistema de control con red neuronal artificial del exoesqueleto ante diferentes tipos de señales de impulso.

Toda la información que fue recopilada en el transcurso de cada procesamiento de las señales de impulso para el control del exoesqueleto por medio de la red neuronal será ubicada en una tabla comparativa para un entendimiento más claro del sistema de control en una placa de desarrollo mediante un método PIL.

Dicha comparación se analiza en dos diferentes aspectos, los cuales son:

- **Análisis de ángulo resultante.** La variación del ángulo en el instante de inicialización, el valor del ángulo en el momento de realizar la primera variación por tiempo de muestreo y por último el valor del ángulo producido por una segunda variación en la señal de impulso.
- **Análisis del tiempo de muestreo.** La alteración existente en el transcurso del tiempo, el cual permite observar la velocidad de muestreo, así como la del procesamiento de cada placa de desarrollo.

### 3.2.1. ANÁLISIS DE ÁNGULO RESULTANTE

En este análisis se realiza la comparación de los ángulos obtenidos en el procesamiento de cada una de las señales de impulso a comparación de la simulación del entorno de Simulink.

Como se observa en la Tabla 2, al poseer un tiempo de muestreo en las placas de desarrollo Arduino Mega 2560 y Arduino Due, estas poseen un valor nulo en el ángulo resultante del procesamiento. Esto a razón de ausencia de valores de entrada con los cuales la red neuronal.

**Tabla 2.** Comparación de ángulo en la inicialización del procesamiento de la señal.

| **Valor del Ángulo en Instante 0 segundos** | | | |
|---|---|---|---|
| | **Señal de Impulso 1** | **Señal de Impulso 2** | **Señal de Impulso 3** |
| **Simulación de Entorno Simulink** | 1°19′44″ | 1°19′44″ | 1°19′44″ |
| **Arduino Mega 2560** | N/A | N/A | N/A |
| **Arduino Due** | N/A | N/A | N/A |
| **Raspberry Pi 3 B+** | 1°18′ | 1°18′ | 1°18′ |

Luego de recibir el primer valor de alteración por parte de la señal de impulso en cada placa de desarrollo, estos generan un valor aproximado del ángulo al valor deseado de cada señal de impulso, como se observa en la Tabla 3.

**Tabla 3.** Comparación de ángulo en la primera alteración por parte del tiempo de muestreo del procesamiento de la señal.

| **Valor del Ángulo en Instante 0.128 segundos** | | | |
|---|---|---|---|
| | **Señal de Impulso 1** | **Señal de Impulso 2** | **Señal de Impulso 3** |
| **Simulación de Entorno Simulink** | **1°19′44″** | **1°19′44″** | **1°19′44″** |
| **Arduino Mega 2560** | **1°19′48″** | **1°19′48″** | **1°19′48″** |
| **Arduino Due** | **1°19′48″** | **1°19′48″** | **1°19′48″** |
| **Raspberry Pi 3 B+** | **1°18′** | **1°18′** | **1°18′** |

En esta se puede observar una inexistencia de diferencia del valor del ángulo resultante en cada placa de desarrollo posterior al procesamiento de cada señal de impulso.

Por tal motivo, se genera la necesidad de observar el valor generado ante una segunda variación en cada una de las señales de impulso para determinar la reacción más eficiente de cada placa de desarrollo. Esto se observa en la Tabla 4, donde se puede denotar una igualdad en la respuesta obtenida del ángulo procesado de las señales de impulso 1 e impulso 2 en las placas Arduino Mega 2560 y Arduino Due.

Esto genera la necesidad del análisis del ángulo obtenido en el procesamiento de la señal de impulso 3, en el cual se observa un comportamiento distinto en cada placa de desarrollo, dando la observación de una respuesta más idónea en la placa de desarrollo Arduino Mega 2560 a comparación del ángulo resultante con la placa de desarrollo Arduino Due.

**Tabla 4.** Comparación de ángulo en la segunda alteración por parte del tiempo de muestreo del procesamiento de la señal.

| **Valor del Ángulo en Instante 0.256 segundos** | | | |
|---|---|---|---|
| | **Señal de Impulso 1** | **Señal de Impulso 2** | **Señal de Impulso 3** |
| **Simulación de Entorno Simulink** | 4°25′58″ | 4°55′40″ | 6°56′45″ |
| **Arduino Mega 2560** | 4°25′48″ | 4°55′48″ | 6°57′ |
| **Arduino Due** | 4°25′48″ | 4°55′48″ | 6°52′48″ |

Por la razón del método PIL directo realizado en placa de desarrollo Raspberry Pi 3 B+, el análisis del resultado a lo largo del tiempo es enfocado al valor resultante del ángulo procesado en su máximo dato obtenido. Este se observa en la Tabla 5, la cual se denota una comparación del ángulo resultante del procesamiento de los diferentes tipos de señales con el ángulo deseado, señalando un comportamiento idóneo del sistema de control embebido en la placa de desarrollo.

**Tabla 5.** Comparación de ángulo máximo generado en la placa Raspberry Pi 3 B+.

| **Valor del Ángulo Máximo Generado en Instantes entre 4 a 5 segundos** | | | |
|---|---|---|---|
| | **Señal de Impulso 1** | **Señal de Impulso 2** | **Señal de Impulso 3** |
| **Ángulo Deseado** | 125°18′ | 125°48′ | 136°30′ |
| **Raspberry Pi 3 B+** | 124° | 123°24′ | 124°18′ |

### 3.2.2. ANÁLISIS DE TIEMPO DE MUESTREO

Este análisis al ser la comparación de la reacción existente por cada alteración de la señal de impulso afectada por el tiempo de muestreo, no se tomará en cuenta la placa de desarrollo Raspberry Pi 3 B+.

Se observa el tiempo de reacción del sistema de control de cada placa de desarrollo ante las diferentes señales de impulso, demostrando un mejor comportamiento en todas estas en la placa de desarrollo Arduino Mega 2560 a comparación de la placa Arduino Due.

Pero a lo largo del tiempo en la segunda interacción, el comportamiento cambia. Esto denotando una mejor reacción a lo largo del tiempo en la placa de desarrollo Arduino Due a comparación de la placa de desarrollo Arduino Mega 2560.

# 4. CONCLUSIONES Y RECOMENDACIONES

# CONCLUSIONES

- Se diseñó un sistema de control con una red neuronal previamente entrenada, para el control de un exoesqueleto de aumento de fuerza de extremidad inferior. La arquitectura de la red neuronal utilizada está definida por 1 capa neuronal conformada por 10 neuronas, encardas de realizar el proceso de control del sistema. Esto es utilizado posterior al tratamiento de la señal de impulso, el cual se encarga de filtrar por distintas etapas, los cuales son el filtrado por pasa bajos, la delimitación de datos por banda y un buffer que permite realizar cada interacción en 0.128 segundos.
- Se utilizó 3 microprocesadores diferentes para el análisis del sistema de control. Este análisis se divide en el costo de cada placa y la velocidad de procesamiento de la señal de impulso, lo cuales se pueden observar en la Tabla 1.
- La comparación de cada placa de desarrollo fue obtenida por el código generado por el método PIL, lo cual permite observar las diferencias existentes de cada una de estas por motivos de la estructura que posee y sobre todo el procesador que está diseñado. Este código se validó por estrategia PIL, demostrando que puede ser implementado en cualquier tipo de placa. Al desarrollar el código programable de las placas, se observó el porcentaje de memoria usada en cada placa utilizada por el método PIL, este porcentaje es: en Arduino Mega 2560 4% equivalente a 10424 bytes de memoria, y en Arduino Due de 4% equivalente a 22952 bytes de memoria interna.
- La prueba MIL/SIL(Modelo en bucle/Software en bucle) se realizó en el software Matlab-Simulink para verificar el procesamiento, análisis de las señales, y el funcionamiento de las redes neuronales artificiales. En la prueba PIL se verificó el funcionamiento de los procesadores de cada una de las placas. Esta verificación se toma en cuenta con los valores que varían a lo largo del tiempo, los cuales son: en Arduino Mega 2560 una variación en el ángulo inicial de 4 segundos, Arduino Due de una variación en el ángulo inicial de 4 segundos y Raspberry Pi 3B+ de una variación en el ángulo inicial de 1 minuto y 4 segundos.

# RECOMENDACIONES

- Se recomienda usar redes neuronales artificiales para el control de un sistema encargado de procesar señales electromiográficas, esto a razón de obtener un resultado ideal y una eficiencia en consumo de recursos para realizar el procesamiento de las señales de entrada. Por tal razón, un control neuronal posee una mejor respuesta a variaciones de distintos tipos y aunque es tiene mayor complejidad en su realización a comparación de otros métodos de control, posee una mejor compatibilidad con las placas.
- Para diseñar un sistema de control neuronal de bajo efecto, no existe diferencia alguna entre las diversas placas. Esto por motivo de no mantener un gran impacto en el procesamiento de diversos tipos de señales de entrada. Por lo cual, al realizar el diseño de un sistema de control para un exoesqueleto de aumento de fuerza es irrelevante la configuración que posee las placas con los diferentes tipos de microcontroladores que conforman su estructura, a razón de no existir una gran variación en el resultado obtenido. Esto da una aclaración al momento de seleccionar una placa específica para el sistema de control, la cual es de motivo económica dando el objetivo de seleccionar la placa de menor coste que existe en el mercado.
- Hay que tomar en cuenta todas las variaciones posibles de microprocesador con la velocidad no son la única propiedad al momento de realizar un sistema de control, esto por motivo de existir más variaciones en la placa a usar. Todos crean diversas variaciones en el procesamiento de una señal de entrada, causando una obtención de un mejor rendimiento del microprocesador.

# BIBLIOGRAFÍA


Akbar iskandar. (2017). *SISTEM KEAMANAN PINTU BERBASIS ARDUINO MEGA.* Macasar: Prodi Informatika Fakultas Teknik Universitas PGRI Semarang.

Alberto López Delis, Andrés F. Ruiz Olaya. (2012). Métodos Computacionales para el Reconocimiento de Patrones Mioeléctricos en el Control de Exoesqueletos Robóticos: una Revisión. *Revista Facultad de Ingenierias de Universidad Antonio Nariño*, 42-59.

ALEXIS GONZALO LAMINGO, LEOPOLDO JAVIER LOOR. (2018). *INVESTIGACIÓN DE MECANISMOS MULTIPLICADORES DE FUERZA PARA EL DISEÑO Y CONSTRUCCIÓN DE UN EXOESQUELETO ROBÓTICO DE EXTREMIDAD SUPERIOR PARA LEVANTAMIENTO DE CARGA.* Quito: DEPARTAMENTO DE CIENCIAS DE LA ENERGÍA Y MECÁNICA.

Arduino. (2017). *Arduino Due.* Italia : Arduino.

Arias, J. L. (2020). *Implementación del sistema de control de un convertidor de potencia usando la técnica PIL para un sistema eléctrico aislado basado en generación solar fotovoltaica.* Sevilla: Dpto. Ingeniería Electrónica de la Universidad de Sevilla.

Aurélei Moyon, Emilie Poirson, Jean-François Petiot. (2018). *Experimental study of the physical impact of a passive exoskeleton on manual sanding operations.* Francia: Elseiver Ltd.

Bonilla Félix Vladimir, Moya Marcelo Javier, Anatoly Vitalyevich Evgeny, Anatolevich Lukyanov, Marín Leonardo Emanuel. (2018). *Modelado y simulación del Robot Mitsubishi RV-2JA controlado mediante señales electromiográficas.* Quito: Universidad UTE.

Christian Fleischer, Gunter Hommel. (2010). *Embedded Control System for a Powered Leg Exoskeleton.* Alemania: Institute for Computer Engineering and Microelectronics, Berlin University of Technology.

Cuesta, T. (2019). *Análisis y Gestión de Riesgos de un prototipo de exoesqueleto de mano para rehabilitación neuro-motora.* Valladolid: UNIVERSIDAD DE VALLADOLID.

Cuevas-Jiménez E., Oliva-Navarro D.A. (2013). *Modelado de filtros IIR usando un algoritmo inspirado en el electromagnetismo.* México: Universidad Nacional Autónoma de México.

Daniel Peralta, Andrés Herrera-Poyatos, Francisco Herrera. (2016). *Un Estudio sobre el Preprocesamiento para Redes Neuronales Profundas y Aplicación sobre Reconocimiento de Dígitos Manuscritos.* Granada:

Departamento de Ciencias de la Computaci´on e Inteligencia Artificial, Universidad de Granada.

Daniel Peralta, Andrés Herrera-Poyatos, Francisco Herrera. (2016). *Un Estudio sobre el Preprocesamiento para Redes Neuronales Profundas y Aplicación sobre Reconocimiento de Dígitos Manuscritos.* Granada: Departamento de Ciencias de la Computación e Inteligencia Artificial.

Franco Núñez, J. K. (2017). *Análisis y diseño de un prototipo de exoesqueleto para la rehabilitación pediátrica de los miembros inferiores, utilizando sistemas embebidos para el control del sistema y la interfaz de usuario.* Guayaquil: UNIVERSIDAD CATÓLICA DE SANTIAGO DE GUAYAQUIL.

Gustavo Aguirre, Ángel Flores, Noé Alba, Carlos Acosta, Ismael Canales. (2015). Control de Señales EMG para el Movimiento de un Brazo Robótico de Tres Grados de Libertad. *Culcyt/ /Mecatrónica*, 6-15.

J.M. Grosso, D. T. (2010). *Diseño conceptual de un exoesqueleto para asistir la rehabilitación de miembro inferior.* Bucaramanga: Universidad Autónoma de Bucaramanga.

Jiri Koziorek, Jaromir Konecny. (2018). *Analysis of dataflows within industrial control system design.* Czech Republic: MATEC Web of Conferences.

Kelson. Denean M., Kim, Sunwook, Nussbaum, Maury A., Srinivasan, Divya. (2019). *EFFECTS OF PASSIVE UPPER-EXTREMITY EXOSKELETON USE ON MOTOR PERFORMANCE IN A PRECISION TASK.* Blacksburg, USA: Virginia Polytechnic Institute and State University.

Manuel Alejandro Chávez Cardona, F. R. (2011). Exoesqueletos para potenciar las capacidades humanas y apoyar la rehabilitación. *Ingeniería Biomédica*, págs 63-73.

Mateusz Sałuch, Daniel Tokarski, Tomasz Grudniewski, Marta Chodyka. (2018). *Raspberry PI 3B + microcomputer as a central control unit in intelligent building automation management systems.* Varsovia: Warsaw University of Technology.

Matlab. (2015). *Lowpass Filter*. Obtenido de MathWorks: https://la.mathworks.com/help/dsp/ref/lowpassfilter.html

Matlab. (2021). *MathWorks*. Obtenido de Bandstop Filter: https://la.mathworks.com/help/dsp/ref/bandstopfilter.html?searchHighlight=bandstop%20filter&s_tid=srchtitle

Matlab. (2021). *MathWorks*. Obtenido de Buffer: https://la.mathworks.com/help/dsp/ref/buffer.html?searchHighlight=buffer&s_tid=srchtitle

Paniagua, C. C. (2018). *Rehabilitación de la marcha con exoesqueleto HAL en lesionados medulares.* Soria: Universidad de Valladolid.

Raíz, P. (2019). *Estudio de pre factibilidad de exoesqueleto robótico para asistir discapacidad y su rehabilitación.* Ibarra: Universidad Tecnica del Norte "UTN".

Ranko Zotovic Stanisic, J. H. (2014). *Estudio de la robustez de los métodos de control para exoesqueletos de miembros inferiores.* Valencia: Universitat Politécnica de Valencia.

Sergio Barrachina, Germán Fabregat, José Vicente Martí. (2015). *Utilizando Arduino Due en la docencia de la entrada/salida.* Castellón de la Plana: Departamento de Ingeniería y Ciencia de los Computadores, Universidad Jaume I.

Veton Këpuska, Humaid Saif Alshamsi. (2016). *Real Time Vehicle Tracking Using Arduino Mega.* Florida: Florida Institute of Technology .

# ANEXOS

# ANEXO 1.

## Código generado por método PIL

**nnEMG.h:**

```
#ifndef RTW_HEADER_nnEMG_h_
#define RTW_HEADER_nnEMG_h_
#include <math.h>
#ifndef nnEMG_COMMON_INCLUDES_
#define nnEMG_COMMON_INCLUDES_
#include "rtwtypes.h"
#endif                                 /* nnEMG_COMMON_INCLUDES_ */
#include "nnEMG_types.h"

#ifndef rtmGetErrorStatus
#define rtmGetErrorStatus(rtm)         ((rtm)->errorStatus)
#endif
#ifndef rtmSetErrorStatus
#define rtmSetErrorStatus(rtm, val)    ((rtm)->errorStatus = (val))
#endif

typedef struct {
  dsp_LowpassFilter_nnEMG_T obj;       /* '<S1>/Lowpass Filter' */
  real_T    GeneratedFilterBlock_FILT_STATE[4];/*    '<S2>/Generated    Filter
Block' */
  real_T Buffer_CircBuf[256];          /* '<S1>/Buffer' */
```

```
  int32_T Buffer_inBufPtrIdx;          /* '<S1>/Buffer' */
  int32_T Buffer_outBufPtrIdx;         /* '<S1>/Buffer' */
  int32_T Buffer_bufferLength;         /* '<S1>/Buffer' */
  boolean_T isInitialized;             /* '<S1>/Lowpass Filter' */
} DW_nnEMG_T;

typedef struct {
real_T IW211_Value[10];
} ConstP_nnEMG_T;

typedef struct {
  real_T sEMG;                         /* '<Root>/sEMG' */
} ExtU_nnEMG_T;

typedef struct {
  real_T angulo;                       /* '<Root>/angulo' */
} ExtY_nnEMG_T;

struct tag_RTM_nnEMG_T {
  const char_T * volatile errorStatus;

 struct {
    struct {
      uint16_T TID[2];
    } TaskCounters;
```

```
  } Timing;
};

extern DW_nnEMG_T nnEMG_DW;
extern ExtU_nnEMG_T nnEMG_U;
extern ExtY_nnEMG_T nnEMG_Y;
extern const ConstP_nnEMG_T nnEMG_ConstP;
 void nnEMG_initialize(void);
 void nnEMG_step(void);
 void nnEMG_terminate(void);

extern RT_MODEL_nnEMG_T *const nnEMG_M;
/*-
 * These blocks were eliminated from the model due to optimizations:
 * Block '<S26>/Subtract min x' : Eliminated nontunable bias of 0
 * Block '<S27>/Add min x' : Eliminated nontunable bias of 0
 */
/*-
 * The generated code includes comments that allow you to trace directly
 * back to the appropriate location in the model.  The basic format
 * is <system>/block_name, where system is the system number (uniquely
 * assigned by Simulink) and block_name is the name of the block.
 *
 * Note that this code originates from a subsystem build,
 * and has its own system numbers different from the parent model.
```

```
 * Refer to the system hierarchy for this subsystem below, and use the

 * MATLAB hilite_system command to trace the generated code back

 * to the parent model.  For example,

 *    hilite_system('nnEMG_codigo/nnEMG')    - opens subsystem
nnEMG_codigo/nnEMG

 * hilite_system('nnEMG_codigo/nnEMG/Kp') - opens and selects block Kp

 * Here is the system hierarchy for this model

 * '<Root>' : 'nnEMG_codigo'

 * '<S1>'   : 'nnEMG_codigo/nnEMG'

 * '<S2>'   : 'nnEMG_codigo/nnEMG/Bandstop Filter'

 * '<S3>'   : 'nnEMG_codigo/nnEMG/Custom Neural Network'

 * '<S4>'   : 'nnEMG_codigo/nnEMG/MATLAB Function'

 * '<S5>'   : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer 1'

 * '<S6>'   : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer 2'

 * '<S7>'   : 'nnEMG_codigo/nnEMG/Custom Neural Network/Process Input 1'

 * '<S8>'   : 'nnEMG_codigo/nnEMG/Custom Neural Network/Process Output
1'

 * '<S9>'   : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer 1/Delays
1'

 * '<S10>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer 1/IW{1,1}'

 * '<S11>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer 1/logsig'

 *   '<S12>'      :  'nnEMG_codigo/nnEMG/Custom  Neural  Network/Layer
1/IW{1,1}/dotprod1'

 *   '<S13>'      :  'nnEMG_codigo/nnEMG/Custom  Neural  Network/Layer
1/IW{1,1}/dotprod10'

 *   '<S14>'      :  'nnEMG_codigo/nnEMG/Custom  Neural  Network/Layer
1/IW{1,1}/dotprod2'
```

```
 * '<S15>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer
1/IW{1,1}/dotprod3'

 * '<S16>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer
1/IW{1,1}/dotprod4'

 * '<S17>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer
1/IW{1,1}/dotprod5'

 * '<S18>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer
1/IW{1,1}/dotprod6'

 * '<S19>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer
1/IW{1,1}/dotprod7'

 * '<S20>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer
1/IW{1,1}/dotprod8'

 * '<S21>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer
1/IW{1,1}/dotprod9'

 * '<S22>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer 2/Delays
1'

 * '<S23>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer 2/LW{2,1}'

 * '<S24>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer 2/purelin'

 * '<S25>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Layer
2/LW{2,1}/dotprod1'

 * '<S26>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Process Input
1/mapminmax'

 * '<S27>'  : 'nnEMG_codigo/nnEMG/Custom Neural Network/Process Output
1/mapminmax_reverse'

 */

#endif                         /* RTW_HEADER_nnEMG_h_ */
```

**nnEMG.cpp:**

```
void nnEMG_initialize(void)

{

  {

    static const real_T tmp[15] = { 0.27464680423556082,
0.012653482990825074,

      0.27464680423556082, 0.41119758021116415, 0.41119758021116415,
0.0,

      0.54449087231528, -0.22711231525780048, 0.54449087231527993,

      0.26951597685224427, 0.26174169735421809, 0.26951597685224449,

      0.73329528925225285, -0.22355899926430814, 0.73329528925225318
};

    static const real_T tmp_0[10] = { -0.74036245885958518,
0.69042402026803962,

      -0.5135656601593398, 0.0, -0.58773778715769165,
0.9656188657140895,

      -0.92047085111043991, 0.42135756799444696, -
0.62723409085482751,

      0.87026567009502576 };

    b_dspcodegen_BiquadFilter_nnE_T *iobj_0;

    int32_T i;

    nnEMG_DW.Buffer_inBufPtrIdx = 128;

    nnEMG_DW.Buffer_bufferLength = 128;

    nnEMG_DW.obj._pobj0.matlabCodegenIsDeleted = true;

    nnEMG_DW.obj.matlabCodegenIsDeleted = true;

    nnEMG_DW.obj.isInitialized = 0;

    nnEMG_DW.obj.NumChannels = -1;

    nnEMG_DW.obj.matlabCodegenIsDeleted = false;
```

```
nnEMG_DW.obj.isSetupComplete = false;
nnEMG_DW.obj.isInitialized = 1;
iobj_0 = &nnEMG_DW.obj._pobj0;
iobj_0->isInitialized = 0;
iobj_0->cSFunObject.P0_ICRTP = 0.0;
for (i = 0; i < 15; i++) {
  iobj_0->cSFunObject.P1_RTP1COEFF[i] = tmp[i];
}

for (i = 0; i < 10; i++) {
  iobj_0->cSFunObject.P2_RTP2COEFF[i] = tmp_0[i];
}
for (i = 0; i < 6; i++) {
  iobj_0->cSFunObject.P3_RTP3COEFF[i] = 0.0;
}
for (i = 0; i < 6; i++) {
  iobj_0->cSFunObject.P4_RTP_COEFF3_BOOL[i] = false;
}
iobj_0->cSFunObject.P5_IC2RTP = 0.0;
iobj_0->matlabCodegenIsDeleted = false;
nnEMG_DW.obj.FilterObj = iobj_0;
nnEMG_DW.obj.NumChannels = 1;
nnEMG_DW.obj.isSetupComplete = true;
iobj_0 = nnEMG_DW.obj.FilterObj;
if (iobj_0->isInitialized == 1) {
```

```
      for (i = 0; i < 10; i++) {
        iobj_0->cSFunObject.W0_ZERO_STATES[i] = iobj_0->cSFunObject.P0_ICRTP;
      }
      for (i = 0; i < 10; i++) {
        iobj_0->cSFunObject.W1_POLE_STATES[i] = iobj_0->cSFunObject.P5_IC2RTP;
      }
    }
  }
}
void nnEMG_terminate(void)
{
  b_dspcodegen_BiquadFilter_nnE_T *obj;
  if (!nnEMG_DW.obj.matlabCodegenIsDeleted) {
    nnEMG_DW.obj.matlabCodegenIsDeleted = true;
    if ((nnEMG_DW.obj.isInitialized == 1) && nnEMG_DW.obj.isSetupComplete) {
      obj = nnEMG_DW.obj.FilterObj;
      if (obj->isInitialized == 1) {
        obj->isInitialized = 2;
      }
      nnEMG_DW.obj.NumChannels = -1;
    }
  }
  obj = &nnEMG_DW.obj._pobj0;
```

```
  if (!obj->matlabCodegenIsDeleted) {
    obj->matlabCodegenIsDeleted = true;
    if (obj->isInitialized == 1) {
      obj->isInitialized = 2;
    }
  }
}
```