ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“Diseño de un sistema electrónico para un exoesqueleto de extremidad superior izquierda basada en procesamiento de señales electromiográfícas e incorporada en una tarjeta de sistemas embebidos con Inteligencia Artificial”

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Electricidad Especialización Electrónica y Automatización Industrial

Presentado por:

Omar Fabricio Alvarado Torres

Eloy Salomón Caín Curicama

GUAYAQUIL - ECUADOR

Año: 2020

DEDICATORIA

El presente proyecto lo dedico a mis padres quienes con su apoyo incondicional y su esfuerzo incansable hicieron que llegue a culminar esta etapa académica, también dedico a mis hermanos y familiares quienes siempre creyeron en mí y se preocuparon de mi instancia lejos de casa.

Omar Alvarado Torres

AGRADECIMIENTOS

Agradezco a Dios primeramente quien es pilar de mi vida y siempre me cuido de todo mal. A mis padres y hermanos que son lo más preciado que tengo y que sin su esfuerzo y trabajo no pudiese haber llegado a estas instancias.

A mis familiares en general que me supieron dar la mano cuando más los necesite.

Y como no a mis amigos de universidad que fueron un gran apoyo y me ayudaron adaptarme al nivel de estudio que se exigía. A todos muchas gracias y que Dios les bendiga por todo su afecto hacia mi persona.

Omar Alvarado Torres

DEDICATORIA

Este proyecto es dedicado especialmente para Dios quien ha sido mi fuerza, esperanza, y mi voluntad, seguidamente la dedico a mi madre quien me ha enseñado a tener coraje ante circunstancias difíciles y de igual manera dedico a mis amigos, Carolina Herrera, Mariam Hinostroza, Kenneth Ramon y Sam Schmitt, quienes me han apoyado en momentos difíciles. Muchas gracias por todas sus enseñanzas y ayuda

Eloy Salomón Caín Curicama

AGRADECIMIENTOS

Mis más sinceros agradecimientos a los Ingenieros Jenny Álava, Dennys Cortez y Ronald Solís quienes ha sido de gran ayuda y soporte en el trascurso de este proyecto. Además, agradezco a Omar Alvarado quien fue mi compañero es este proyecto.

Eloy Salomón Caín Curicama

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; *Omar Fabricio Alvarado Torres y Eloy Salomón Caín Curicama* damos mi nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Omar Fabricio Alvarado Torres

Eloy Salomón Caín Curicama

EVALUADORES

Ing. Dennys Dick Cortez Álvarez

PROFESOR DE LA MATERIA

Ing. Ronald David Solís Mesa

PROFESOR TUTOR

RESUMEN

El presente proyecto tiene como objetivo realizar el diseño de un sistema electrónico para un exoesqueleto de extremidad superior izquierda basada en procesamiento de señales electromiograficas e incorporado en una tarjeta de sistemas embebidos con inteligencia artificial. Los seres humanos somos propensos a sufrir daños cerebrales que conllevan a una discapacidad neuromotora temporal o de por vida, para la cual necesitan de rehabilitación física para recuperar la movilidad de sus miembros afectados. El tiempo de recuperación va acorde a los estragos causados por la enfermedad y por el constante trabajo en rehabilitación, es ahí donde muchos investigadores buscan la manera de aportar en buscar alternativas que permitan un acortar el tiempo de recuperación, desde el punto de vista de la ingeniería se ha propuesto dispositivos que se acoplen al miembro afectado y permita simular sus movimientos teniendo buenos resultados. Sin embargo, estos dispositivos en el mercado actual son de altos costos siendo un impedimento al momento de adquirirlo para la mayoría de los pacientes. Se puede encontrar exoesqueletos de bajo costo, pero con reducidas funciones mayormente son de funcionalidad mecánica. El enfoque del proyecto es realizar un sistema electrónico y control el cual que permita complementar a estos exoesqueletos de bajo costo y le de cierta autonomía similar a los exoesqueletos robotizados de altos costos.

El sistema consta de sensores de señales electromiográficas, una tarjeta embebida que contiene algoritmo con redes neuronales, baterías, drivers para motores y finalmente los motores todos estos elementos tendrán su respectivo análisis de costos como su funcionalidad.

Palabras Clave: red neuronal, tarjeta embebida, señales electromiograficas.

ABSTRACT

The present project aims to carry out the design of an electronic system for a left upper extremity exoskeleton based on electromyographic signal processing and incorporated into an embedded system board with artificial intelligence. Human beings are prone to brain damage that leads to a temporary or lifelong neuromotor disability, for which they need physical rehabilitation to regain the mobility of their affected limbs. The recovery time is according to the ravages caused by the disease and by the constant work in rehabilitation, it is there where many investigators look for the way to contribute in looking for alternatives that allow a shortening of the recovery time, from the point of view of the engineering it has been proposed devices that are coupled to the affected member and allow to simulate its movements having good results. However, in the current market, these devices are of high cost, which is an impediment for most patients when acquiring them. Low-cost exoskeletons can be found, but with reduced functions they are mostly of mechanical functionality. The focus of the project is to develop an electronic and control system that will complement these low-cost exoskeletons and give them a certain autonomy like high-cost robotic exoskeletons.

The system consists of electromyographic signal sensors, an embedded card containing algorithm with neural networks, batteries, drivers for motors and finally the motors all these elements will have their respective cost analysis as their functionality.

Keywords: *neural network, embedded card, electromyographic signals.*

ÍNDICE GENERAL

ABREVIATURAS

EMG	Electromiografía
DC	Corriente directa
GPU	Unidad de procesamiento grafico
Opamp	Amplificador operacional
LPF	Filtro paso bajo
HPF	Filtro paso alto
BIAS	Corriente de Polarización
CMRR	Factor de rechazo en modo común
GPIO	Entrada y salida de propósito general
ADC	Conversor análogo digital
PCB	Placa de circuito impreso
ACV	Accidente cerebrovascular

SIMBOLOGÍA

Hz	Hertz
mV	milivoltio
H	Henrio
nH	nano henrio
mA	miliamperio
µV	microvoltio
dB	Decibel

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

Las enfermedades cerebrovasculares es la segunda causante de muerte y a su vez el tercer lugar como factor que causa discapacidad a nivel mundial (Neurología, 2017), siendo estas discapacidades pueden ser temporales o a largo plazo, en tanto la mano es el miembro más afectado y su probabilidad de recuperar su uso funcional depende netamente de la gravedad que se haya presentado en la enfermedad.

Esta discapacidad hace que la persona presente problemas al realizar actividades diarias como sujetar objetos, comer o escribir. Por consiguiente, es necesario que estos pacientes realicen terapias de rehabilitación física en centros especializados para su respectivo tratamiento, los cuales consisten en electroestimulación, estimulación con tecnología robótica, tecnología inalámbrica y realidad virtual. Los resultados esperados con estas terapias son restaurar la función de la mano, mejorar motricidad, pero sobre todo mejorar su calidad de vida (Clinic, 2019).

Como se mencionó previamente la rehabilitación puede ser actividades físicas normales o actividades físicas asistidas por tecnología, siendo esta ultima el enfoque del proyecto. El desarrollo de dispositivos en los últimos años se ha ido incrementando con el fin de encontrar uno que solvente la necesidad del paciente.

Los exoesqueletos son ortesis activas que actúan como dispositivos biomecánicos superficiales que recubren una parte del cuerpo para aumentar sus capacidades o a su vez para compensar el trabajo de un sistema musculoesquelético que fue afectado por problemas neurológicos. (Cruz-Martínez, 2018).

1.1 Descripción del problema

El aumento de enfermedades cerebrovasculares en los últimos tiempos ha motivado a que se busque implementar dispositivos que ayuden en la rehabilitación de estos pacientes, siendo desarrollados muchos modelos ya sea por las grandes marcas Biomédicas o por desarrolladores sin fines de lucro.

El diseño de un exoesqueleto demanda tiempo y conocimientos avanzados tanto en mecánica como electrónica, siendo este último un impedimento para que se culmine con éxito los prototipos, por esta razón hemos planteado acaparar con la parte electrónica donde el proyecto consiste en el diseño del sistema electrónico es decir todo lo que requiere para que un dispositivo funcione (motores con sus respectivos drivers, sensores EMG, tarjeta de control y sistema de alimentación para estos elementos). Para que estos elementos funcionen es necesario que exista una señal que active al sistema, es por eso por lo que se plantea el procesamiento de señales Electromiografícas que son captadas en los músculos donde se da el proceso de contracción y relajación en el antebrazo izquierdo, estas señales son procesadas en una tarjeta Raspberry Pi 4 donde se procesa las señales y se implementa la red neuronal para finalmente controlar los motores que

van en los dedos de la mano para que se puedan dar los movimientos de flexión y extensión, durante la rehabilitación del paciente.

1.2 Objetivos

1.2.1 Objetivo General

Diseño de un sistema electrónico para un exoesqueleto de mano izquierda, incorporando algoritmos de inteligencia artificial en una tarjeta embebida, para que active los actuadores del prototipo y permita realizar ejercicios de rehabilitación.

1.2.2 Objetivos Específicos

1. Diseño de un algoritmo para la red neuronal artificial que será implementada en la tarjeta “Raspberry pi 4” y que mediante esta se podrá tener control con los actuadores.
2. Usar datos de una publicación científica de mediciones realizadas con sensores EMG para luego analizar y extraer sus parámetros principales que permitan tener un método de desarrollo del algoritmo.
3. Realizar un análisis electrónico de los componentes que comprenden en el sistema tales como motores con sus respectivos controladores, sensores EMG, tarjeta Raspberry pi 4 y baterías de alimentación. Además de realizar el diseño esquemático de conexiones de estos elementos.
4. Seleccionar los dispositivos que incluyen en el sistema donde el factor principal que estos sea su bajo costo.

1.3 Marco teórico

1.3.1 Exoesqueleto para miembro superior

Un exoesqueleto tal como se muestra en la figura 1.1 es una estructura mecánica, robótica o servo armadura que se asemeja a un armazón externo, el cual cubre parcial o totalmente el cuerpo de una persona con el fin de aumentar sus capacidades físicas o brindar asistencia para personas que perdieron su movilidad. Estos dispositivos pueden ser estructuras activas como pasivas, es decir pueden o no contener actuadores que permitan dan movimientos autónomos al dispositivo.

Gran parte de los exoesqueletos son desarrollados para la medicina y se acoplan al cuerpo con sistemas inteligentes de procesamiento de señales bioeléctricas para su debido control.

Figura 1.1 Exoesqueleto robotizado para mano. Hertfordshire (2015)

1.3.2 Sensor EMG(Electromiografía)

La electromiografía es una rama de la bioelectrónica que tiene como objetivo captar y procesar señales eléctricas producidas por la contracción muscular en un ser humano.

Los sensores EMG en la actualidad siguen siendo de suma importancia en la biomédica, muchos de estos sensores se desarrollan como instrumentos de estudio y para áreas clínicas, estos sensores son utilizados para la captación de contracciones musculares y puestos en prótesis robotizadas como también en exoesqueletos de rehabilitación.

EMG es sinónimo de electromiograma donde la contracción muscular se mide con potenciales eléctricos, los valores generados en una contracción se encuentran alrededor de los 100μV. Estas tensiones son muy atenuadas debido al tejido cutáneo y tejido muscular, para lo cual su diseño electrónico se compone de 3 etapas tales como preamplificación (Coras, 2012), filtrado y amplificación. Las referencias consideradas en ciertos músculos como el bíceps alrededor de 2.1mV en amplitud y con una frecuencia que va desde 2 hasta 500 Hz. (Coras, 2012)

1.3.2.1 Amplificador de instrumentación medica AD620

El AD620 es un integrado de instrumentación medica de bajo costo y alta precisión que solo requiere una resistencia para poder establecer su ganancia que van desde 1 hasta 10000.

Genera poca potencia en el rango aproximado de 1.3mA y 50μV como máximos comprende de rangos adaptables para energizarse desde (±2.3 a ±18V) el cual fácilmente se puede hacer uso de pilas o baterías portátiles. Cuenta con una compensación de 06 µV / ° C bajo ruido y baja corriente de polarización de entrada que hace que sea muy adecuado en el ámbito medico como para la adquisición de señales bioeléctricas. En la figura 1.2 se puede observar una aplicación típica del AD620 donde se lo utiliza como parte de un sensor ECG. (Mouser Electronics, 2020)

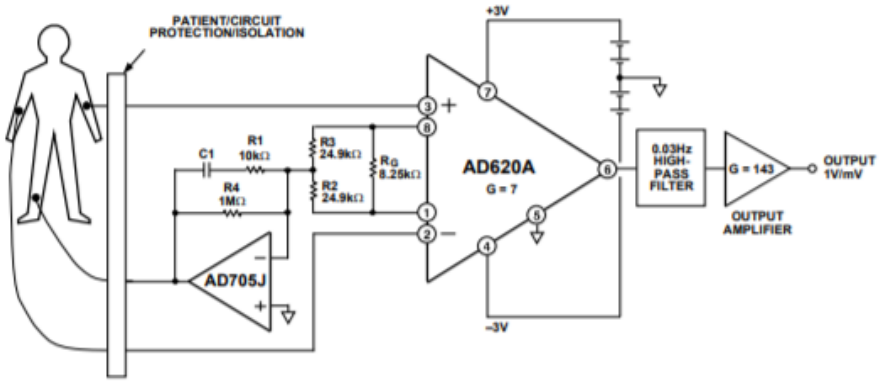


Figura 1.2 Representación del uso del amplificador en aplicaciones médicas. Device (2011)

1.3.2.2 Filtro pasa bajo activo.

Los filtros pasan bajos activos de primer orden hace referencia a su nombre el cual solo permite el paso de frecuencias bajas y atenúa a las frecuencias altas. Este circuito comprende por elementos pasivos tales como un capacitor, resistencias y de un Opam, este último elemento permite que se comporte de manera diferente a los filtros pasivos. El Opam además de filtrar las señales entrantes permite amplificar la señal, en la figura 1.3 se muestra el diseño del filtro de primer orden donde se caracteriza porque en su entrada se encuentra una resistencia y un capacitor puesto a tierra.

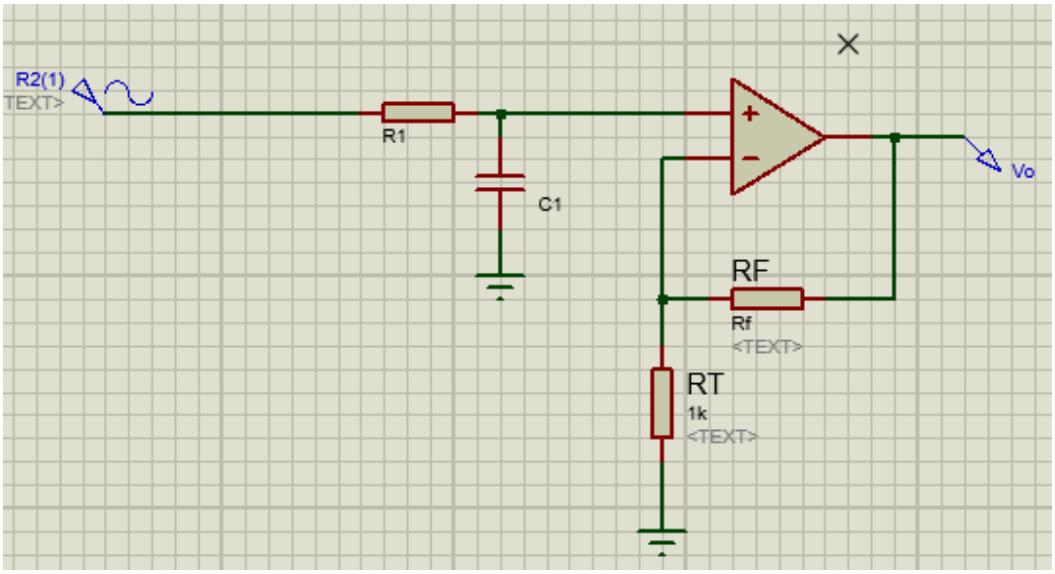


Figura 1.3 Filtro pasa bajo activo de primer orden

Su salida se determina mediante las ecuaciones:

$$V_o = \left(1 + \frac{Rf}{RT}\right).V_i \quad [V] \tag{1.1}$$

$$A_v = 1 + \frac{Rf}{RT} \tag{1.2}$$

El capacitor es el elemento determinante para el diseño de filtros ya que su reactancia capacitiva se opone al paso de corriente en función de la frecuencia.

$$X_C = \frac{1}{2\pi f C_1} \quad [\Omega] \tag{1.3}$$

$$f_{OH} = \frac{1}{2\pi R_1 C_1} \quad [Hz] \tag{1.4}$$

1.3.2.3 Filtro pasa alto pasivo.

Los filtros pasan alto pasivo de primer orden permite el paso de frecuencias altas y atenúa a las frecuencias bajas y además sirve de filtro DC, es decir elimina el offset que se genera en la salida de cualquier amplificador previamente. Este circuito

comprende por elementos pasivos tales como un capacitor y una resistencia como se muestra en la figura 1.4.

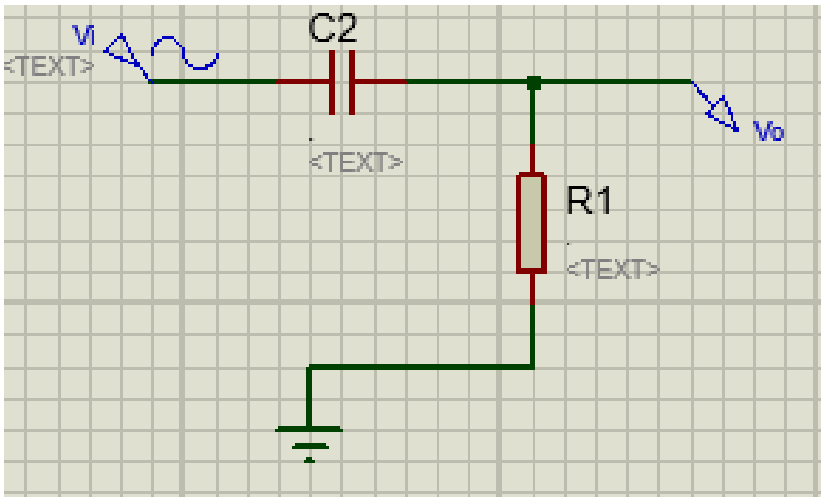


Figura 1.4 Filtro pasa alto pasivo de primer orden

Su salida se determina mediante las ecuaciones:

$$V_o = \frac{V_i R}{R1+Z_C} \quad [V] \tag{1.5}$$

$$\frac{V_o}{V_i} = \frac{R}{R+\frac{1}{jwC}} \tag{1.6}$$

$$f_{OL} = \frac{1}{2\pi R_1 C_2} \text{ [Hz]} \tag{1.7}$$

1.3.2.4 Filtro Notch gyrator.

El circuito gyrator es un convertidor de impedancia activo que, al usar un condensador como componente reactivo, el gyrator transforma de impedancia capacitiva a inductiva. Al usar esta configuración para diseñar un filtro Notch o comúnmente conocido como filtro rechaza banda se usa para eliminar los ruidos

que aparecen en la inducción eléctrica propia de una red 60Hz, en la figura 1.5 se observa que este filtro compone de dos Opams siendo uno de ellos que tiene retroalimentación negativa.

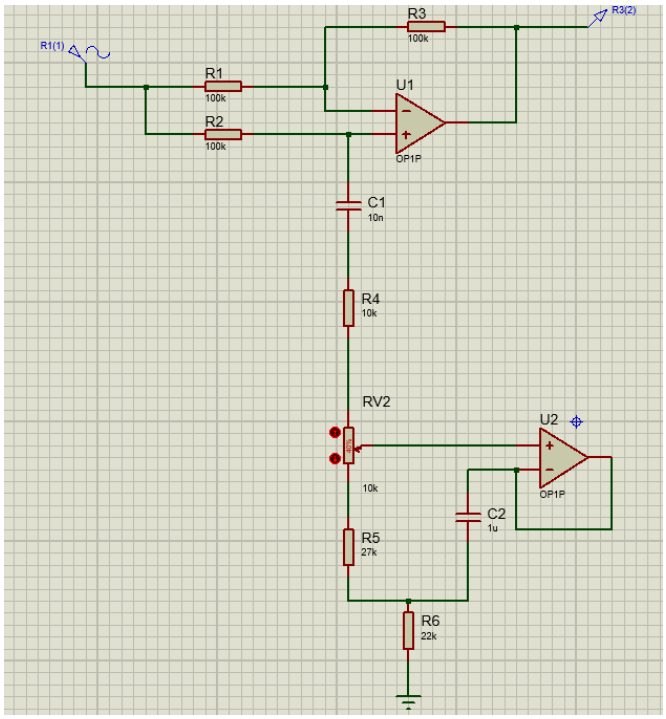


Figura 1.5 Filtro Notch

Su salida se determina mediante las ecuaciones:

$$R = R_5 + R_6 \qquad [\Omega] \qquad (1.8)$$

$$L = C_2 * R_5 * R_6 \qquad [H] \qquad (1.9)$$

$$f = \frac{1}{2\pi\sqrt{LC_1}} \qquad [Hz] \qquad (1.10)$$

1.3.3 Señales Electromiografícas

Las señales electromiograficas (EMG) son señales eléctricas producidas por los músculos durante el proceso de relajación y contracción. Estas características musculares son aprovechadas para el control mediante una computadora o alguna tarjeta en desarrollo con amplia capacidad procesadora, con el fin de crear interfaces de comunicación entre un usuario y la máquina de alguna índole. Una interfaz comúnmente usada es la que se da entre una persona con cierto grado de discapacidad física y un equipo de rehabilitación, siendo las contracciones musculares las que activen a esos equipos.

Estas señales se originan analógicamente, es decir se crean en ciertos instantes de tiempo y para poder trabajar con ellas se las debe convertir ya sea en señales digitales o discretas, una de las técnicas que se utiliza para no perder información es tener en rango de frecuencias.

“El teorema del muestreo afirma que para muestrear una señal sin perder información y por tanto poder reproducirla perfectamente partiendo de la señal discreta, se debe muestrear a una frecuencia de al menos el doble de la frecuencia fundamental de la señal analógica” (Rubio, 1999)

1.3.4 Tarjeta de sistemas embebidos.

Los sistemas embebidos se caracterizan por tener partes computarizadas con componentes especiales conocidos como “microcontroladores” que conforman el cerebro del sistema. El conjunto de esos integrados es llamado

“microprocesadores” en la cual incluyen interfaces de entrada/salida en un mismo chip además de tener una interfaz externa para monitorear y diagnosticar los resultados. Los sistemas embebidos se pueden programar directamente en el lenguaje ensamblador del microcontrolador o en lenguaje del microprocesador

1.3.5 Actuador lineal eléctrico

Un actuador lineal eléctrico es un dispositivo dinámico integrado por un motor y un tornillo el cual convierte movimiento giratorio producido en el eje del motor en movimiento lineal trasmitido al vástago del pistón. En el mercado se puede encontrar actuadores a gran escala como también a pequeñas escalas, es así como para este proyecto se enfatiza en el actuador servo-eléctrico lineal quien cuenta con las mismas características previamente explicado, pero con la suplantación del motor por un servomotor, el cual hace que se pueda controlar más parámetros como velocidad, desplazamiento y empuje.

1.3.6 Controlador de motores

Un controlador es un pequeño dispositivo que amplifica corriente, su función es tomar la señal que sale del controlador que tiene baja corriente y seguidamente convertirla en una señal alta de corriente y que esta pueda conducir al motor.

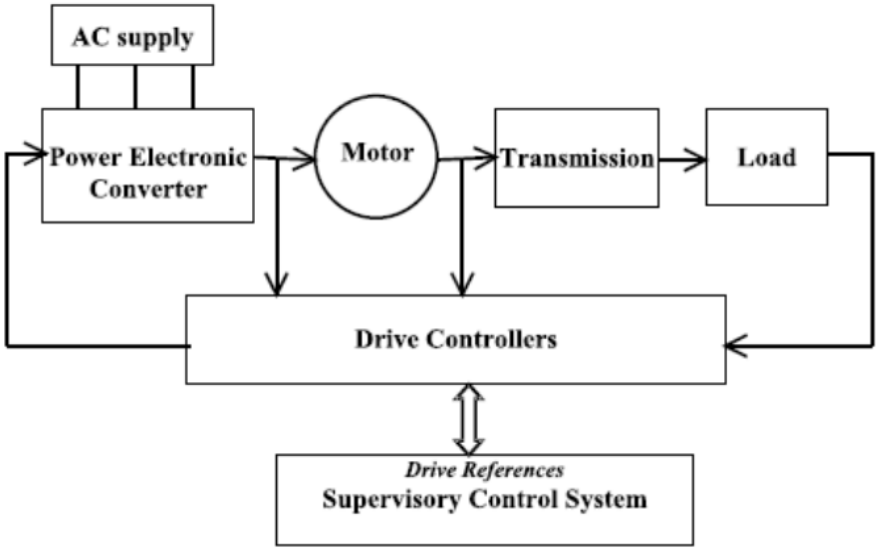


Figura 1.6 Diagrama de bloques de un típico drive de motores. Betz (2011)

1.3.7 Redes neuronales artificiales

Las redes neuronales son modelos predictibles que funcionan de manera similar al sistema nervioso, su forma se asemeja especialmente a una neurona. Se las conoce como algoritmos propios de Machine Learning que aprenden a partir de datos colocados en sus entradas y al pasar por sus enlazadas capas que procesan los datos con el fin de dar una salida acorde la necesidad del programador.

En la figura 1.7 se puede observar una red neuronal donde las capas se dividen en tres partes: capas de entrada, capas ocultas o cuerpo y capas de salida, cuando estas se conectan suelen hacerse entre variables (o ponderaciones). En cuanto más capas contenga la red más compleja es esta, y a su vez también será capaz de resolver funciones más complejas. (2017, 2020)

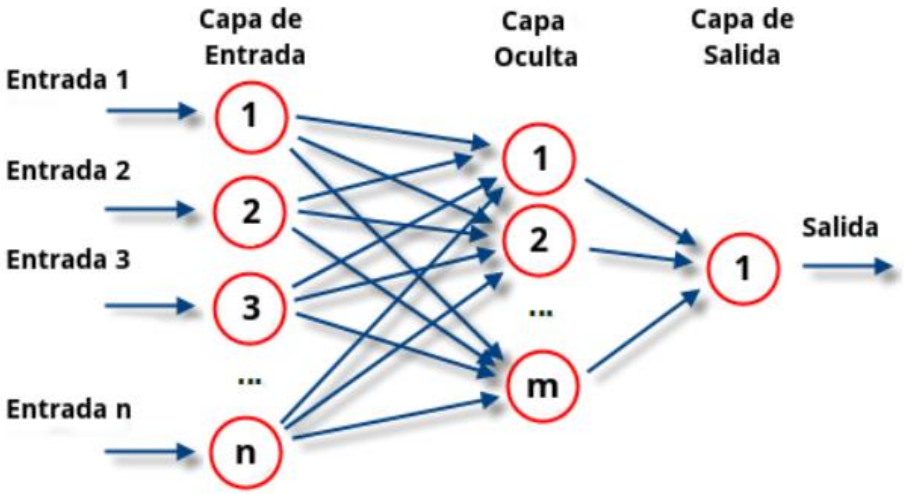


Figura 1.7 Representación gráfica de una red neuronal. Esacademic (2010)

En una red neuronal quienes permiten manipular los datos son llamados “pesos”, estos permiten alterar la función de la red neuronal con el fin de reducir el error al momento de entrenamiento de la red neuronal.

CAPÍTULO 2

2. METODOLOGÍA

En este capítulo se hizo un análisis de los componentes que intervienen en el sistema electrónico con el fin de obtener un diseño de alto nivel y que sea de fácil adquisición para los usuarios para lo cual se plasmó en la figura 2.1, siendo este el procedimiento que se planeó para este capítulo, el cual es ir analizando etapa por etapa.

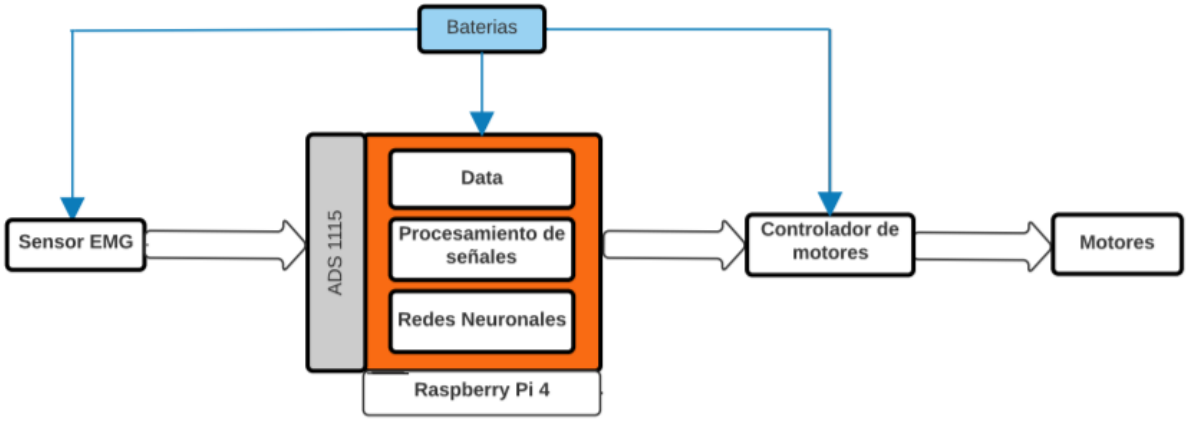


Figura 2.1 Diagrama de proceso del proyecto

2.1.1 Diseño del sensor EMG

El sensor EMG constituye uno de los componentes primordiales para este proyecto siendo el puente que une el medio físico con el digital, por lo cual fue necesario analizar su diseño y más específicamente sus parámetros de filtrado.

En el mercado se puede encontrar diferentes tipos de sensores y cada uno cuenta con sus propias características, entre los más económicos se puede encontrar:

Dispositivo	Características	Precio
MyoWare™Muscle Sensor	*Vs:* +3.3V a +5V *Vsignal:* 0 a +Vs *Impedancia de entrada:*110GΩ *Is:* 9mA a 14mA *CMRR:* 110 *Input Bias:* 1pA	$60.00
Gravity: Analog EMG Sensor by OYMotion	*Vs:* +3.3V a +5V *Vsignal:* ±1.5mV *Electrode Connector:* PJ-342 *Module Connector:* PH2.0-3P *Incluye filtro digital*	$57.00

Tabla 2.1 Características de sensores comerciales

En la tabla 2.1 se muestra dos clases de sensores comerciales, cada uno con sus propias características, además de su costo. Para el presente proyecto se tomó la decisión de construir un propio sensor EMG considerando que los componentes electrónicos que comprenden el circuito son de uso común, por lo cual se pudo encontrar en cualquier tienda y esto permite un ahorro al momento de facturar el proyecto.

A continuación, se mostrará el diseño del sensor. Para lo cual se partió de los rangos de frecuencia que normalmente trabajan los sensores EMG.

Tipo de Señal	Rango de Frecuencias
Complejo P, QRS, T	0.05-100Hz
Monitor ECG	0.67-40 Hz
Electro grama Intracardiaco	10-1kHz
EMG Superficial	2-500Hz
Potenciales de acción de potencia motoras	5-10kHz

Tabla 2.2 Rango de frecuencias de señales bioeléctricas

Como se muestra en la tabla 2.2 las señales del sensor EMG superficial se encuentran en rango de 20 a 500Hz siendo la referencia para el diseño de los filtros.

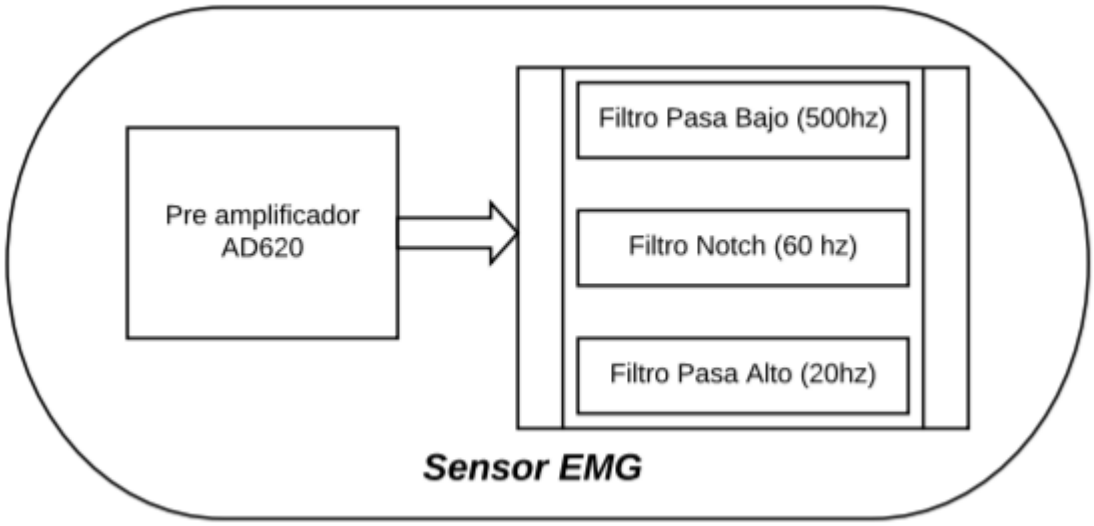


Figura 2.2 Etapas a ser consideradas en el diseño del sensor

Parámetros	Valor	Características
Ganancia pre-amplificación	1000	CMRR=130 dB BW= 12kHz
Frecuencia de corte (LPF)	500Hz	Filtro activo de primer orden
Frecuencia de corte (HPF)	20Hz	Filtro pasivo de primer orden
Frecuencia rechazada.	60Hz	Filtro Notch (Tipo Gyrator)

Tabla 2.3 Características de cada etapa del sensor EMG

En la figura 2.2 referencia a las etapas que conforman al sensor y que fueron configurados acorde los valores que se muestran en la tabla 2.3.

2.1.2 Pre-amplificación

El AD620 es usado para esta etapa y a continuación en la tabla 2.4 se especifica las características de este elemento.

Características	**Detalles**	**Rangos**
Bajo voltaje Offset	La salida Dc del amplificador tiende a ser baja y permite la saturación de los demás amplificadores	<50 µV

Baja corriente de polarización (BIAS) de entrada	Cuando es bajo el consumo de corriente en la entrada, hace que el electrodo no cambie su composición electroquímica.	<2nA
Alto CMRR	Relación que hace entre la ganancia en modo diferencial y modo común.	>130dB
Rango de energización.	Permite elegir los rangos de alimentación eléctrica acorde requiera el usuario.	±2.3 a ±18V
Encapsulado	Depende del diseño electrónico que se realice.	DIP y SOIC

Tabla 2.4 Características del amplificador de instrumentación medica AD620

En el capítulo I se menciona que el amplificador puede tener un rango de ganancia hasta de 10000 donde se lo asigna acorde el valor resistencia R_G que es colocada, un valor muy alto de ganancia puede afectar al rechazo en modo, por lo que se consideró mantener una ganancia estandarizada para evitar cualquier ruido inesperado.

Para determinar la ganancia del amplificador se aplica la ecuación 2.1:

$$G = \frac{49.4k\Omega}{R_G} + 1 \qquad (2.1)$$

Si se requiere una ganancia de G= 1000 se reemplaza los datos en la ecuación 2.1 y se halla el valor de la resistencia.

$$R_G = \frac{49.4k}{G-1} \qquad [\Omega]$$

$$R_G = \frac{49.4k}{1000-1} \qquad [\Omega]$$

$$R_G = 49.44 \qquad [\Omega]$$

Se obtuvo un valor R_G de 49.44 Ω pero cabe indicar que comercialmente no existe resistencias de este valor por el cual el más cercano es de 47 Ω, teniendo como CMRR 120 dB y respuesta de frecuencia dentro de los 12KHz.

2.1.2.1 Filtro pasa bajo

Para obtener una frecuencia de corte de 500 Hz se usó la ecuación 2.2 y se asignó valores a los componentes que no cuentan con un rango considerable de valores, como por ejemplo el capacitor.

C=100nF

$f_{OH} = 500Hz$

$$f_{OH} = \frac{1}{2\pi R_1 C_1} \qquad (2.2)$$

$$R_1 = \frac{1}{2\pi f_{OH} C_1}$$

$$R_1 = \frac{1}{2\pi * 500 * 100n}$$

$$R_1 = 3.18K \ [\Omega]$$

Como es de conocimiento la señal de salida del sensor es de tipo análogo y consecuentemente para la siguiente etapa es el conversor análogo digital ADS 1115, los puertos tienen un rango de valores a receptar y están entre (0-1.8) V. Fue necesario asignar una ganancia a este filtro, considerado un valor de 28.

Si $A_v = 28$ y R_T=1k

$$A_v = 1 + \frac{R_f}{R_T}$$

$$28 = 1 + \frac{R_f}{1k}$$

$$R_f = 27k\ [\Omega]$$

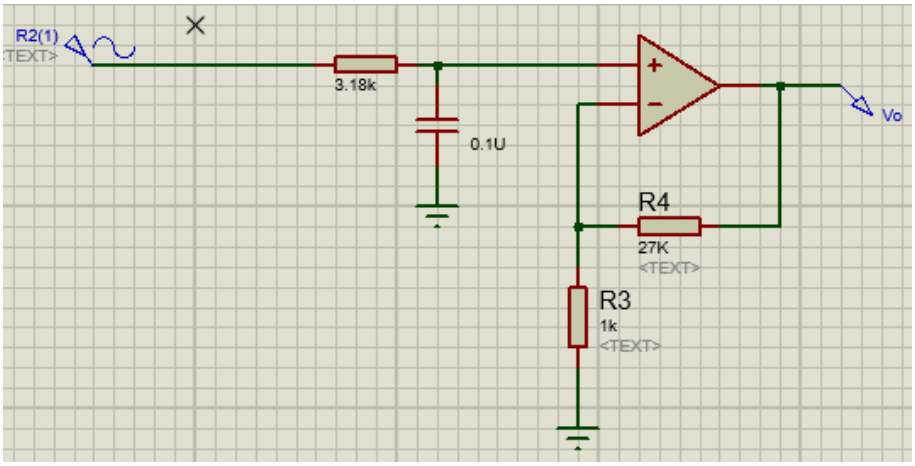


Figura 2.3 Esquemático Filtro pasa bajo con valores

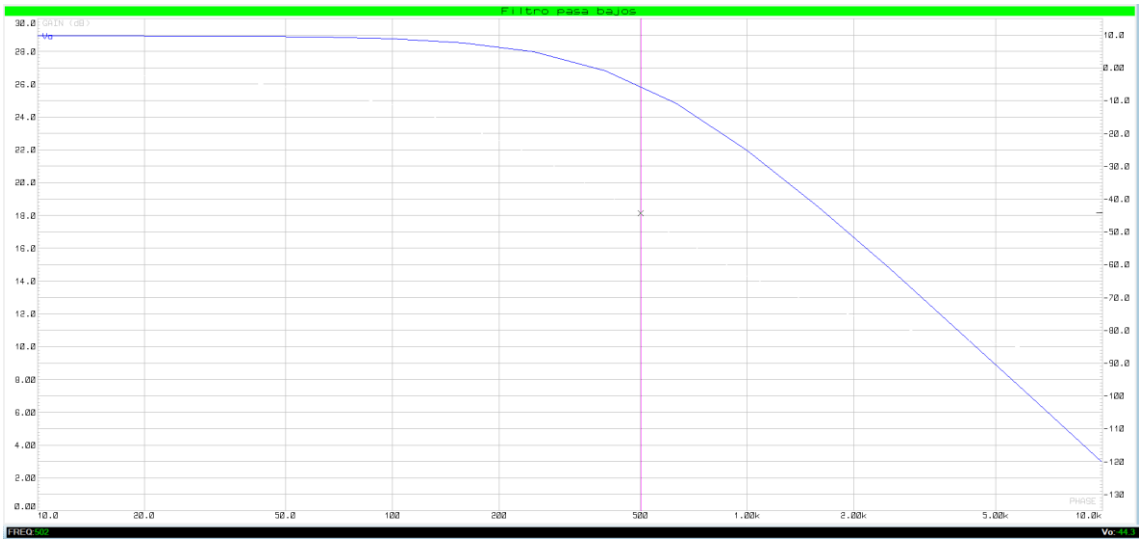


Figura 2.4 Respuesta en análisis de frecuencia LPF.

Realizado los correspondientes cálculos con las fórmulas ya mencionadas anteriormente se asignó valores al filtro pasa bajo como se muestra en la figura 2.3 y en la figura 2.4 la respuesta de frecuencia que obtuvo siendo visible que a menor de 500Hz este trabaja bien.

2.1.2.2 Filtro Notch gyrator

El filtro Notch es imprescindible en este sistema ya que permite eliminar el ruido que se genera debido a la inducción eléctrica es decir los 60Hz, se seleccionó un filtro tipo gyrator debido a que este permite convertir la impedancia reactiva del capacitor a una inductiva. La configuración gyrator también es conocida como “inductores simulados” (Shenoi, 1965) ya que un inductor normal tiene la característica de tener baja impedancia cuando registra una señal continua siempre y cuando este sea ideal. Pero en la vida real no existe un elemento ideal ya que sus elementos de construcción generan diferentes impedancias que al final afectan, además de que los gyrator son

configurados para trabajar hasta los 30KHz de manera óptima. En resumen, se necesitó una configuración gyrator con el fin de evitar que se cree altas impedancias al sistema, teniendo suficiente con la que aparece al conectar los electrodos.

Para su configuración se requirió la ecuación 1.10 en el cual se dio como dato principal la frecuencia de corte igual a 60Hz.

$$f = \frac{1}{2\pi\sqrt{LC_1}}$$

$$60 = \frac{1}{2\pi\sqrt{LC_1}}$$

$$\sqrt{LC_1} = \frac{1}{2\pi * 60}$$

$$LC_1 = 7.03x10^{-6}$$

Si $C_1 = 10nF$

$$L = \frac{7.03x10^{-6}}{10x10^{-9}}$$

$$L = 703 \qquad [H]$$

Dada la ecuación 1.9:

Si $R_5 = 27k$

$$R_6 = \frac{L}{C_2 * R_5}$$

$$R_6 = \frac{703}{1x10^{-6} * 27k}$$

$$R_6 = 22k \ [\Omega]$$

Para la etapa de amplificación se utilizó una configuración con resistencias iguales ya que se desea obtener una ganancia unitaria.

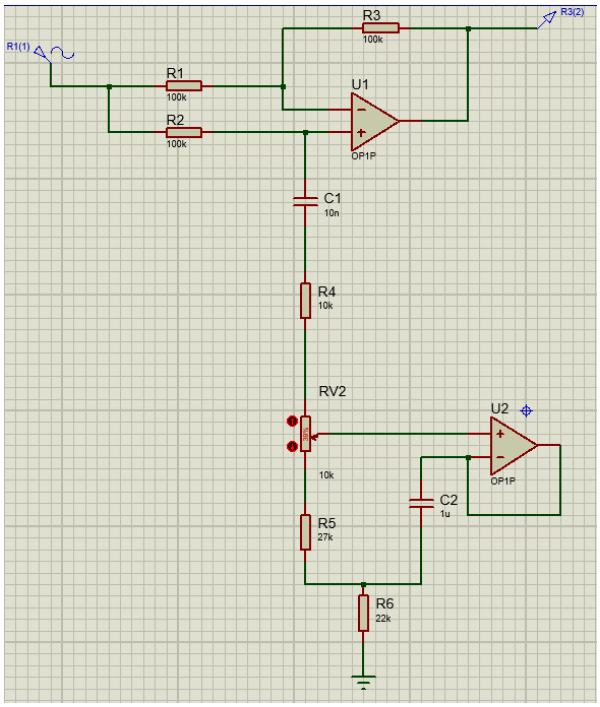


Figura 2.5 Configuración del filtro Notch con valores

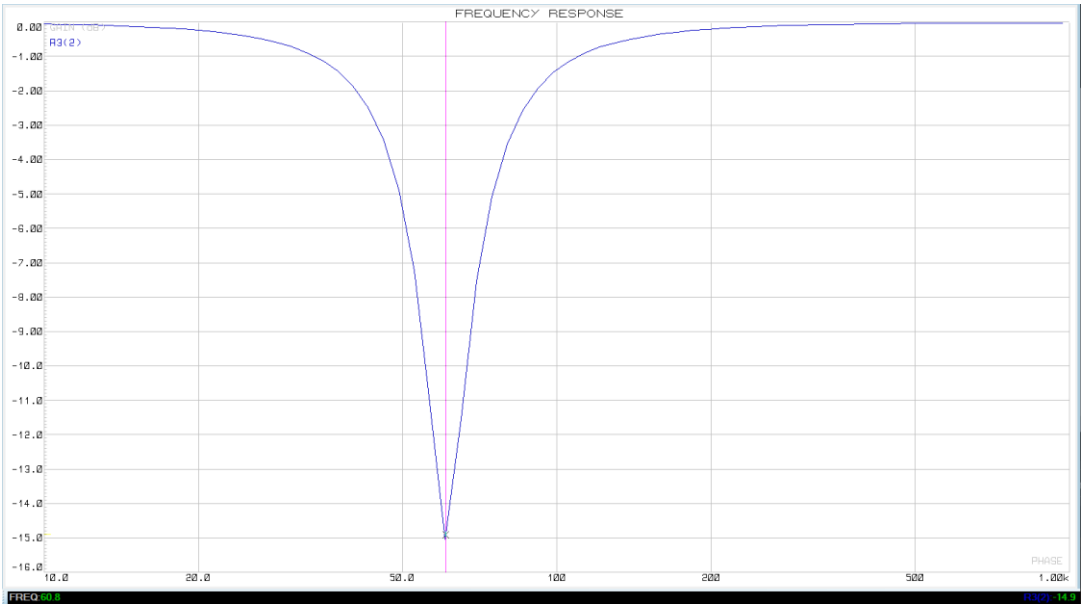


Figura 2.6 Respuesta en análisis de frecuencia del filtro Notch

En la figura 2.5 se muestra el filtro Notch con sus respectivos valores simulados siendo relevante indicar que la ganancia de este filtro es unitaria y en la figura 2.6 la respuesta de frecuencia que se obtuvo donde si rechaza los 60Hz.

2.1.2.3 Filtro pasa alto

Para esta etapa se implementó un filtro pasivo de primer orden con frecuencia de corte de 20 Hz, acorde a las ecuaciones 1.4 que corresponden a este circuito, se obtuvo los siguientes valores para su configuración.

$$f_{OL} = \frac{1}{2\pi R_1 C_2} \quad \text{[Hz]}$$

Si $f_{OL} = 20Hz$ y $C_2 = 100nF$

$$f_{OL} = \frac{1}{2\pi R_1 C_2}$$

$$R_1 = \frac{1}{2\pi f_{OL} C_2}$$

$$R_1 = \frac{1}{2\pi(20)(100n)}$$

$$R_1 = 80K \ [\Omega]$$

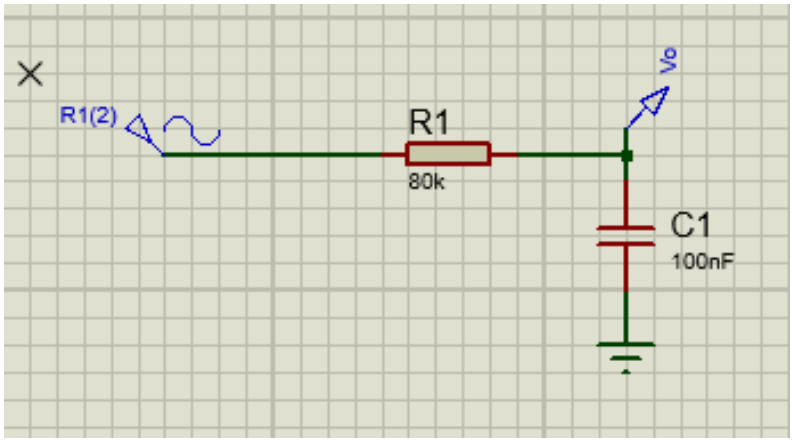


Figura 2.7 Configuración del filtro pasa alto con valores

.

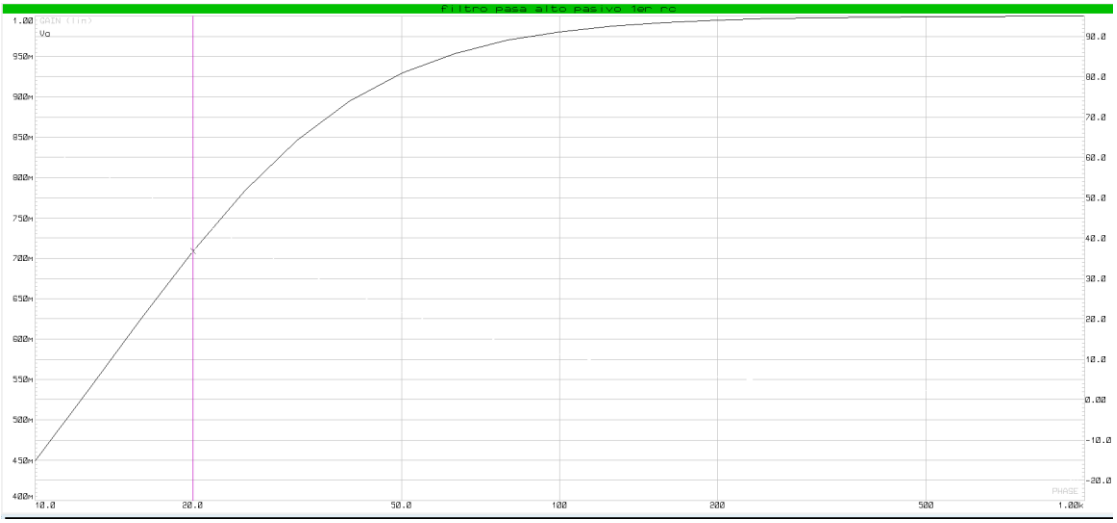


Figura 2.8 Respuesta en análisis de frecuencia filtro pasa alto

En la figura 2.7 se muestra los valores asignados a esta configuración sencilla y en la figura 2.8 la respuesta de frecuencia que se obtuvo la cual a los 20Hz disminuye considerablemente la señal tendiéndose a eliminarse.

Cabe recalcar que se debió seleccionar filtros de primer orden con el fin de evitar incluir muchos componentes ya que estos suelen ser causantes de ruido térmico, además de disminuir costos.

2.3.1 Implementación de la tarjeta embebida.

Al hablar de una tarjeta embebida se hace referencia a su hardware y software, elementos como microprocesadores quien aporta la capacidad de cómputo al sistema, el tipo de memoria se interna o externa, los puertos de ethernet, HDMI, VGA, GPIO e I2C son los parámetros que hacen que se incline a un modelo especifico sin olvidar el lenguaje de programación.

A continuación, se muestra las tarjetas más usadas y sus características.

	FPGA nano 10	**Raspberry Pi 4 B**	**Beaglebone Ai**
Procesador	Dual Cortex-A9	ARM Cortex-A72	Dual Arm Cortex-A15.
RAM	1GB DDR3	4 GB LPDDR4 SDRAM	1GB de RAM y 16GB eMMC
USB	USB OTG x 1	2 x USB 2.0 2 x USB 3.0	USB x 2
Audio	----------	HDMI/RCA	microHDMI
Video	HDMI	2 puertos micro HDMI (hasta 4K a 60 FPS), 2-lane MIPI DSI display, 2-lane MIPI CSI	Subsistema IVA-HD (codificación y decodificación 4K @ 15fps para H.264, 1080p60

		camera, 4-pole stereo audio y video	
Ethernet	Gigabit	Gigabit	Gigabit
I/O	40 pins GPIO x2, Arduino expansión x1	40 pin header x1	69 GPIO, LCD, GPMC, MMC1, MMC2, 7 AIN, 4 temporizadores, 4 puertos seriales, CAN0
Sistema Operativo	Linux, Windows	Linux	Android, Linux, Windows, Cloud9, CE, etc
Entorno	Lenguaje en C, Quartus	Linux, IDLE, OpenEmbedded, QEMU, Scratchbox, Eclipse	Python, Scratch, Linux, Eclipse, Android ADK
Opiniones	Para su programación demanda altos conocimientos de programación	Su programación es de fácil accesibilidad y cuenta con mucha	Su programación es de software libre pero la poca información de esta tarjeta hace que sea requiera

	y un software con licencia.	información, además de contar con software libre.	alargar el tiempo de investigacion.
Costo	130	100	130

Tabla 2.5 Comparación de tarjetas embebidas

Con el análisis realizado en la tabla 2.4 se eligió la tarjeta Raspberry pi 4 quien en la figura 2.9 muestra sus principales partes funcionales.

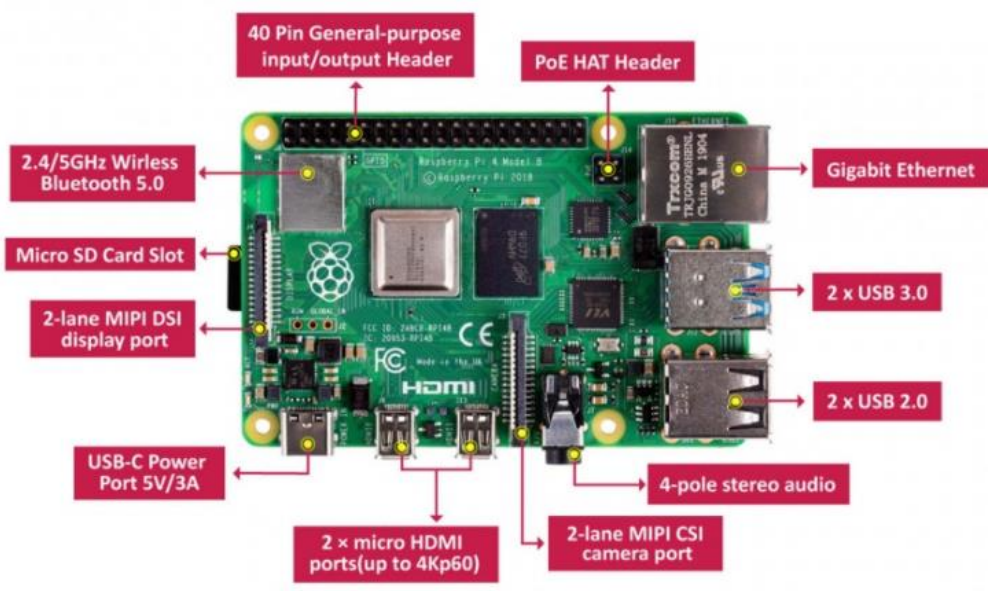


Figura 2.9 Raspberry pi 4 y sus partes funcionales. Raspberry Pi 4 (2020)

2.3.2 Data

Dado las circunstancias que se desarrolló este proyecto se adquirió una base de datos de un repositorio con el nombre “Electromyogram (EMG) Feature Reduction Using Mutual Components Analysis for Multifunction Prosthetic Fingers Control” (Rami

Khushaba, 2020) en el cual se encontró datos correspondientes a 8 personas de las cuales fueron 6 hombres y 2 mujeres con edades comprendidas entre 20-35 años, estos fueron reclutados para realizar ciertos movimientos de los dedos ya sean simples o combinados. El procedimiento que realizaron fue el de conectar 8 electrodos superficiales colocados sobre la región muscular que genera movimientos a la mano. Las señales obtenidas fueron pasadas previamente por etapas de amplificación con una ganancia igual a la sugerida en el sensor EMG de 1000.

Fue usado un convertidor análogo-digital de 12 bits quien permitió realizar un muestreo de la señal a 4000 Hz mediante el software “Delsys EMGWorks”, por lo cual las señales pasaron por un filtro digital de rango entre 20-450 Hz. Siendo varios movimientos realizados pero los que más se enfatizaron fueron los individuales que constan de flexión y extensión del pulgar, índice, medio, anular y meñique adicionando el movimiento del puño.

2.3.3 Procesamiento de señales

2.3.3.1 Selección de software

Al realizar el procesamiento de datos se puede encontrar varios métodos y softwares que permiten realizar esta acción. Sin embargo, en el ámbito de desarrollo académico se destacan dos herramientas Matlab y Python.

Matlab con sus paquetes de librerías permiten tener un entorno de laboratorio ideal para tratamiento digital de señales siendo utilizado en las instituciones de pre y

postgrado. Sin embargo, para instalar este software es necesario adquirir una licencia y tener una capacidad considerable de memoria en el PC, siendo razones para considerar otros ámbitos de programación.

A continuación, en la figura 2.10 se realizó una comparación entre estas dos herramientas, donde se recalca sus funciones principales tales como se puede indicar Matlab permite realizar desde operaciones básicas, operaciones de matrices, implementación de algoritmos hasta creaciones de interfaces de usuarios (GUI) además que permite comunicaciones con dispositivos tal como Arduino y Raspberry Pi. También cuenta con paquetes con sus propios IDE. Matlab conlleva ciertas ventajas en la programación científica en comparación de las demás herramientas, pero Python ha crecido y ha incorporado sus propias interfaces como se los puede enunciar Numpy, Scioy, Matplotlib además de su IDE.

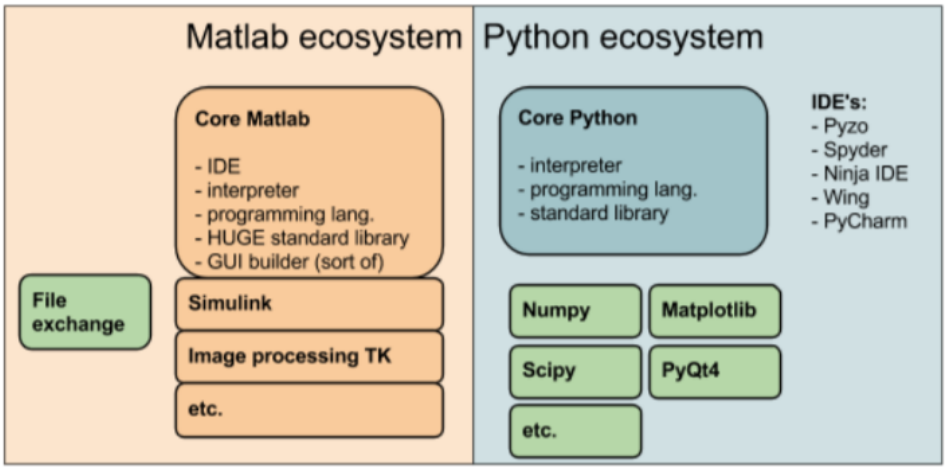


Figura 2.10 Comparación entre las dos herramientas de programación. Ozgur (2017)

Una de las ventajas más sobresaliente de Python es que es un software libre y gratuito, permitiendo que se cree paquetes y herramientas por sus usuarios.

Por los detalles indicados se eligió Python para realizar el procesamiento de señales siendo uno de los detalles más importantes que el usuario podrá manipular el código ingresado en la tarjeta.

2.3.3.2 Procesamiento de señales

Antes de entrar al algoritmo con los datos adquiridos para el entrenamiento se debe verificar que estos se encuentren de forma adecuada. Es por eso en algunos algoritmos para redes neuronales son muy sensibles a la escala de datos, dando alta o baja prioridad aquellos que se representan con valores mayores. También se presentan valores nulos o desconocidos en el conjunto de datos que a la larga traen problemas y se debe tener una solución.

Por último, se debe tener en cuenta que los algoritmos necesitan valores numéricos para poder ser entrenados, por lo que obliga a codificar los posibles valores categóricos que están en el conjunto de datos.

2.3.3.3 Escalado de datos

El escalonado es un método que facilita el funcionamiento en muchos algoritmos de Machine Learning usados en Data Science, para esto se aplicó la “normalización de las variables de entrada” al algoritmo. Si se habla de normalizar se

hace referencia a comprimir o extender los valores de las variables con el fin que se encuentren en un rango definido.

Sin embargo, si se llega a realizar un mal normalizado o una elección inapropiada al método, este puede afectar a los datos y con ello el análisis. (Morante, 2018)

2.3.3.3.1 Escalado estandard (Standard Scaler)

Es una alternativa para el escalado de variables, se basa en el artificio de que a cada dato se le reste la media de la variable y se le divida por la desviación típica.

La fórmula del escalado estándar es:

$$X_{normalized} = \frac{X - X_{media}}{X_{stddev}} \quad (2.3)$$

Las dos fórmulas estadísticas (media y desviación standard) tienen ciertas características que hacen que esta función no sea aplicable para todos los datos, es decir son sensibles a valores anómalos ya sean muy grandes o pequeños. Otro incidente podría ser donde los datos no lleguen a normalizar entre 0-1 y ya con las anomalías que aparecieron, es necesario eliminarlas, pero en el proceso se eliminarían datos que son importantes. Dejando a la base de datos incompleta y con opción a que falle todo el proyecto. Es por lo que este método se lo utiliza para señales estables.

2.3.3.3.2 Escalado de variables (Feature Scaling o MinMax Scaler)

A diferencia del Standard Scaler el escalador MinMaxScaler transformar las características escalándolas a un rango personalizado por el programador y si no así se asigna por defecto (0,1). (interactivechaos, 2020)

La fórmula del escalado MinMax Scaler es:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (2.4)$$

El escalado mediante esta función permite comprimir los datos de entrada entre sus límites empíricos (máximos y mínimos de la variable), permitiendo que los datos tomen nuevas referencias y sea más notable sus variaciones, es por eso que (Morante, 2018) resalta que esta función es ideal para señales inestables tal cual como se caracteriza una señal bioeléctrica.

Realizado este previo análisis se optó por usar loa función de MinMax Scaler.

2.3.4 Red Neuronal

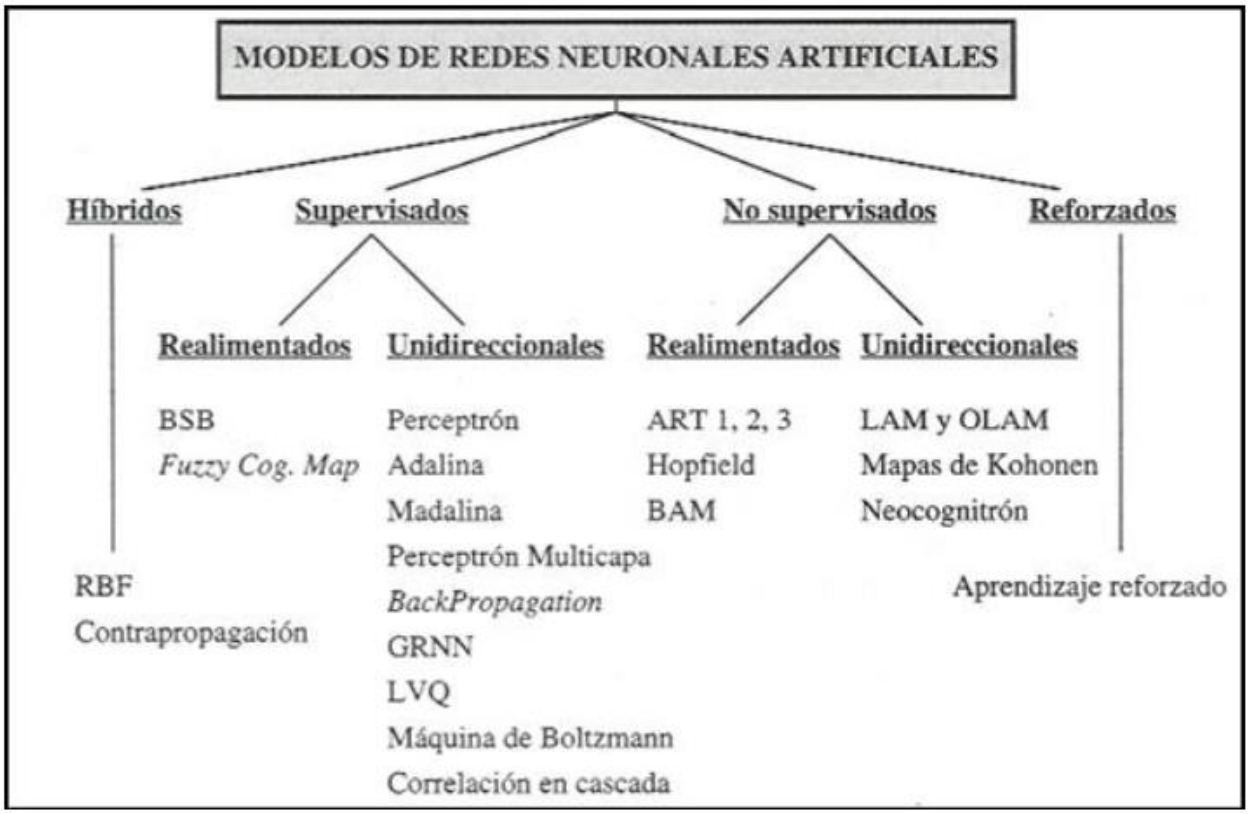


Figura 2.11 Clasificación de los modelos de redes neuronales

En la figura 2.11 se observa los tipos de algoritmos más conocidos para la implementación de una red neuronal.

Para el vigente proyecto se utilizó el tipo supervisado/clasificador ya que el objetivo del proyecto es predecir el tipo de dato que ingresa y que se active la correspondiente salida. Para realizar el algoritmo se hizo uso de las librerías que han sido implementadas por Python para el ámbito de Inteligencia Artificial, para lo cual a continuación se enunciara las herramientas más importantes:

2.3.4.1 Numpy

Numpy es un paquete que provee Python cuando aparecen datos tipo listas, enteros, puntos flotantes y necesitan pasar por cálculos científicos y arreglos multidimensionales de alta eficiencia.

Los arreglos de Numpy permiten tener una mayor eficiencia al momento de usar la memoria

2.3.4.2 Librería Pandas

La librería panda (su nombre deriva de panel data) hace referencia a un conjunto de datos estructurados multidimensionales. Esta librería tiene como función el proporcionar estructuras de datos y funciones de alto nivel como manipular tablas numéricas y series temporales, estas se basaron en el array multidimensional de NumPy.

Para importa la librería se la realiza con el alias “pd” pero previamente debe importar Numpy, como se muestra en el siguiente ejemplo:

```
In [1]: import pandas as pd
        import numpy as np

In [2]: s = pd.Series([-2, 0, 3, 6])
        print(s)

        0   -2
        1    0
        2    3
```

Figura 2.12 Estructura de programación librería Pandas

En la figura 2.12 se observa un típico ejemplo de llamada de las librerías donde numpy reconoce la operación a realizar y panda se encarga de resolverlo.

2.3.4.3 Funciones de activación

Las funciones de activación son las encargadas de devolver una salida a partir de un valor de entrada, los rangos de salidas más utilizados están entre [0,1], para ello se tiene buscan funciones donde las derivadas sean simples permitiendo minimizar el proceso computacional sin embargo esta función tiene como desventajas su lenta convergencia, pero compensa en el buen rendimiento en su última capa.

A continuación, se muestra los tipos de funciones:

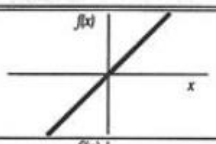
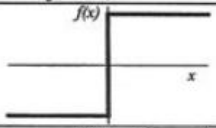
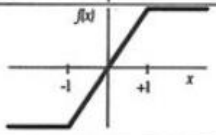
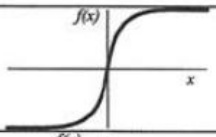
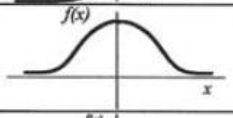
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Figura 2.13 Tipos de funciones de Activación de una Red Neuronal

En la figura 2.15 se observa los diferentes tipos de funciones para la cual selecciono la función Sigmoidea, debido a que se requiere una salida binaria de [0,1] además los datos de entrada no tienen una linealidad constante es por eso que la función sigmoidea permite una entrada de cualquier valor y en su salida la convierte en datos binarios.

2.4.1 Controlador de motores

Al momento de elegir un controlador se investigó ciertas maneras de implementarlos ya sea partir de sus componentes bases como el integrado puente h “l293d” o a su vez tarjetas comerciales que realizan esta función.

La tarjeta Shield “TB6612FNG” es el complemento ideal para controlar motores DC, este permite controlar hasta 2 dispositivos a una corriente de 1.2A (3.2A pico). El circuito integrado TB6612FNG consta de dos puentes H. En la figura 2.16 se observa a la tarjeta mostrando que su construcción es con tecnología SMD.

Especificaciones:

- Chip: TB6612FNG (Toshiba)
- Canales: 2 (soporta 2 motores DC o 1 motor PAP)
- Voltaje de Potencia (VMOT): 5V - 15V
- Voltaje Operación (VCC): 2.7V - 5.5V
- Capacidad de corriente: 1.2A (picos de hasta 3A)
- Potencia máxima disipada: 1W
- Posee diodos internos de protección

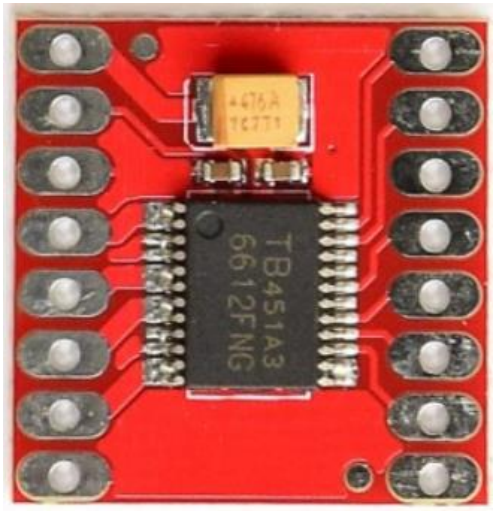


Figura 2.14 Controlador de motores TB6612FNG. Naylamp (2020)

2.5.1 Selección de motores

Al momento de elegir los motores se puede tener muchas alternativas por la variedad que existen en el mercado. Sin embargo, para el uso de exoesqueletos se reduce a un grupo concreto donde implican servomotores de los cuales se pueden tener servomotores lineales y rotativos. Al hablar de servomotores rotativos se hace referencia al rpm que este puede generar, a su forma cilíndrica y a relación entre su fuerza y dimensiones, en el cual mientras más fuerza se requiere más grandes son estos.

Al comparar con los servomotores lineales esto cambia debido a sus características cuentan con una simplicidad mecánica, rigidez elevada, elevadas aceleraciones y velocidades, alta precisión, efectiva repetibilidad y lo más importante su reducido tamaño permiten que sean los indicados para trabajar con prótesis y exoesqueletos. (sapiensman, 2020)

Uno de los modelos que se consideraron son los Actuadores lineales micro L12 de la marca “actounix”, son micro servos creados para trabajar con lógica CMOS y es plug and play, son compatible con la mayoría de los receptores RC. Además de trabajar con la mayoría de las tarjetas programables se puede adquirir acorde la longitud de final de carrera que se requiera. A continuación, se muestra las especificaciones más detallados. (Actuonix, 2020)

L12 Specifications

Gearing Option		50:1	100:1	210:1
Peak Power Point		17N @ 14mm/s	31N @ 7mm/s	62N @ 3.2mm/s
Peak Efficiency Point		10N @ 19mm/s	17N @ 10mm/s	36N @ 4.5mm/s
Max Speed *(no load)*		25mm/s	13mm/s	6.5mm/s
Max Force *(lifted)*		22N	42N	80N
Back Drive Force *(static)*		12N	22N	45N
Stroke Option	10 mm	30mm	50mm	100mm
Mass	28 g	34 g	40 g	56 g
Repeatability *(-I,-R,-P&LAC)*	±0.1 mm	±0.2 mm	±0.3 mm	±0.5 mm
Max Side Load *(extended)*	50N	40N	30N	15N
Closed Length *(hole to hole)*	62mm	82mm	102mm	152mm
Potentiometer *(-I, -R, -P)*	1kΩ±50%	3kΩ±50%	6kΩ±50%	11kΩ±50%
Voltage Option		6VDC		12VDC
Max Input Voltage		7.5V		13.5V
Stall Current		460mA		185mA
Standby Current *(-I/-R)*		7.2mA		3.3mA
Operating Temperature		-10°C to +50°C		
Potentiometer Linearity		Less than 2.00%		
Max Duty Cycle		20 %		
Audible Noise		55dB @ 45cm		
Ingress Protection		IP-54		
Mechanical Backlash		0.2mm		
Limit Switches *(-S)*		Max. Current Leakage: 8uA		
Maximum Static Force		200N		

Figura 2.15 Especificaciones de los motores L12.

Figura 2.16 Servomotor lineal. Actuonix (2020)

En la figura 2.15 se puede apreciar sus especificaciones de este motor donde se destaca lo ya mencionado, además de alimentarse con un voltaje y corriente fácil de acoplar al controlador de motores y finalmente en la figura 2.16 se observa la forma que estos motores tienen.

2.6.1 Baterías

Para seleccionar las baterías se procedió a verificar el consumo de energía de cada elemento y así seleccionar la batería.

Dispositivo	***Voltaje***	***Corriente***
2 sensores EMG	9V	100mA
Raspberry pi 4	5V	3000mA
Controladores de motores	5V	1200 mA
5 motores lineales	6-7.5 V	2300mA
Consumo Total		6100mA

Tabla 2.6 Análisis de consumo de los dispositivos

Como se muestra en la tabla 2.6 el consumo del sistema electrónico se encuentra en los 6100mA por consiguiente se procedió a buscar en el mercado un powerbank que permita tener una durabilidad considerable dando como resultado un powerbank de 40000mA.

Teniendo en cuenta la capacidad del powerbank se puede calcular el tiempo de funcionamiento de los componentes:

$$tiempo = \frac{capacidad\ de\ bateria}{consumo\ total} \qquad (2.5)$$

$$tiempo = \frac{40000mA}{6100mA}$$

$$tiempo = 6.55\ horas\ Aprox.$$

Mediante la ecuación 2.5 se pudo calcular el tiempo de duración de la batería siendo relevante el tiempo de duración de 6.55 horas si este se encuentra en un uso constante.

2.6.2 Características de la batería.

En la tabla 2.7 se observa las características detalladamente de la powerbank a igual se observa la figura 2.17 donde se destaca sus múltiples salidas con voltajes de 5V/9V/12V a 3A de corriente.

Interfaz de salida:	DC, USB/Micro USB, Tipo C	**Interfaz de entrada:**	DC
Tipo de batería:	Batería de polímero de litio	**Lugar del origen:**	Guangdong, China
Peso:	1155g	**De entrada:**	19V/2A
Outport:	Salida DC 5V/8,4 V/9V 9V 3A 12V/16V/20V 4.7A	**TYPE-C de salida:**	5V/9V/12V 3A (puede cargar MacBook)
Salida USB:	5V/9V/12V 3A (admite Carga Rápida)	**certificación:**	CE ROHS FCC MSDS
Tiempo de carga:	6 ~ 7 horas	**Dimensión:**	195*150*28mm

Garantía:	12 meses	**Características:**	Carga rápida 3,0

Tabla 2.7 Características de la powerbank para el sistema

Figura 2.17 Powerbank de 40000mA y sus puertos. Shenzhen Nejifu (2015)

2.7 Conversor ADC

Se debe mencionar que la tarjeta Raspberry Pi 4 no cuenta con entradas para señales analógicas siendo necesario recurrir a dispositivos que permitan realizar esta conversión tales como Arduino o integrados específicamente creado para este proceso. Por razones de tamaño, consumo de energía y costo se procedió a usar un integrado específicamente creado para realizar la conversión análoga/digital, se trata del ADS1115 el cual proporción un ADC de 16 bits de los cuales 15 bits son para medición y un último para el signo. Su comunicación con la placa se realiza mediante el puerto I2C. Contiene una PGA que permite trabajar con ganancias variables que

van desde 0.256 V hasta 6.14V siendo relevante al momento de trabajar con señales que tienen una tensión menor a 5V, en la figura 2.18 se observa las conexiones que se debe realizar con respecto a la Raspberry. (Llamas, Ingenieria,informatica y diseño, 2017)

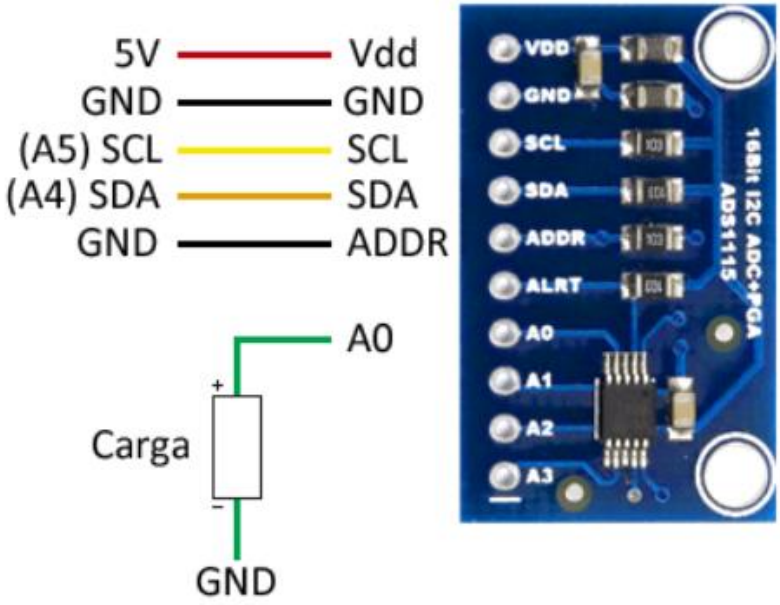


Figura 2.18 Conversor ADS1115 y sus partes. Llamas (2016)

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

3.1 Sensor EMG

Realizado el análisis de diseño en el anterior capitulo, se procedió a realizar el respectivo esquemático y PCB del circuito.

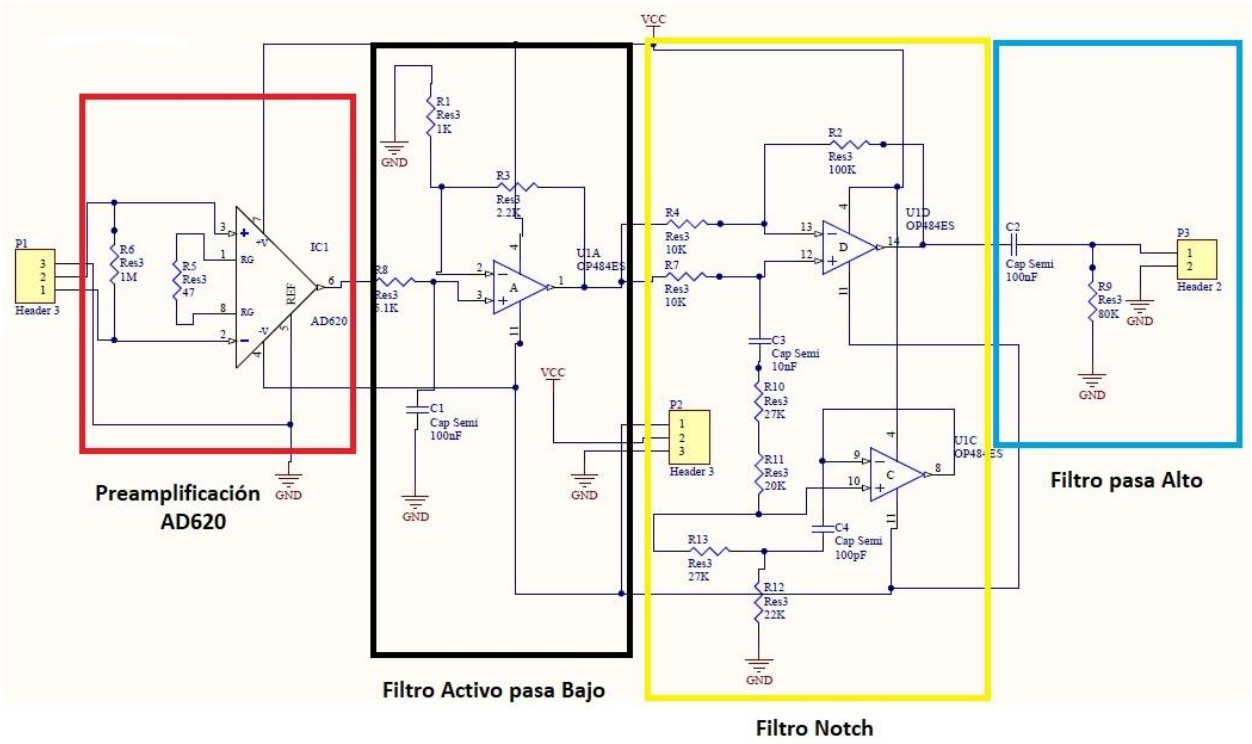


Figura 3.1 Diseño esquemático del sensor EMG

En la figura 3.1 se muestra el diseño completo del sensor, donde consta de 4 etapas como se indica en la figura 2.2. Este circuito contiene su respectiva simulación en respuesta de frecuencia figura 3.2.

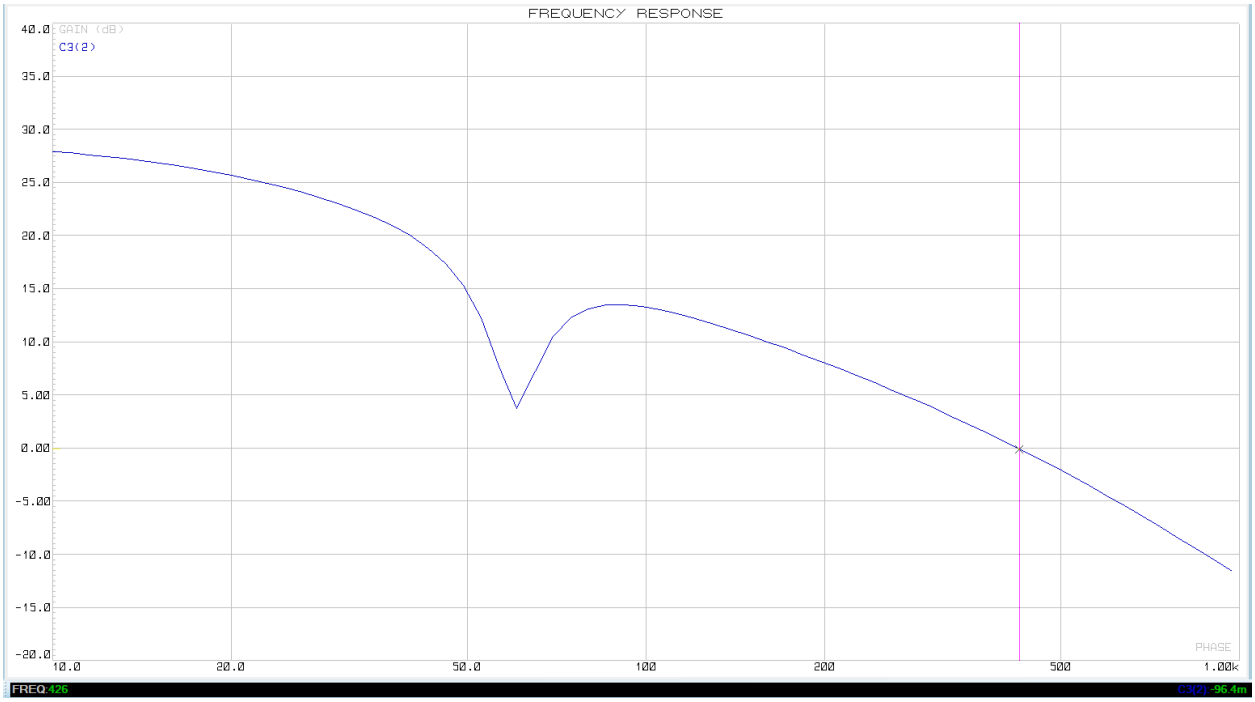


Figura 3.2 Respuesta de frecuencia del sensor EMG

En la figura 3.2 se puede observar ciertas acotaciones que es necesario resaltar siendo la primera, las ganancias asignadas para cada etapa si se cumple se inicia con un valor de 27 con un filtro pasa alto activo de primer orden. Continuando con un filtro Notch gyrator y finalmente con un filtro pasa bajo pasivo donde la frecuencia de corte es 426Hz.

Mediante esta grafica se puede justificar por qué se implementó un preprocesamiento de datos en la tarjeta embebida. Como se mencionó los filtros son de primer orden y esto hace que no tenga un buen filtrado. Si se deseara tener un excelente filtrado sería necesario diseñar filtros de orden superior y eso conllevaría una gran cantidad de componentes, haciendo que el costo sea elevado.

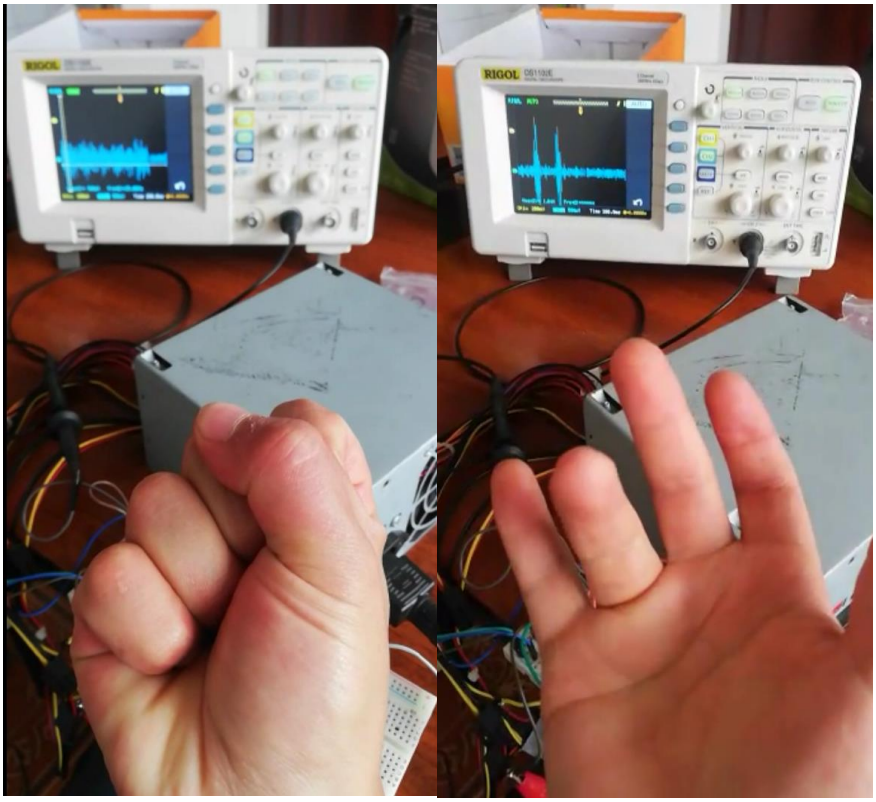


Figura 3.3 Prueba de funcionamiento del sensor EMG implementado en protoboard

En la figura 3.3 se observa el funcionamiento del sensor EMG donde se realizó movimiento tanto de flexión y extensión de cada dedo como también del puño.

3.1.1 Diseño de PCB

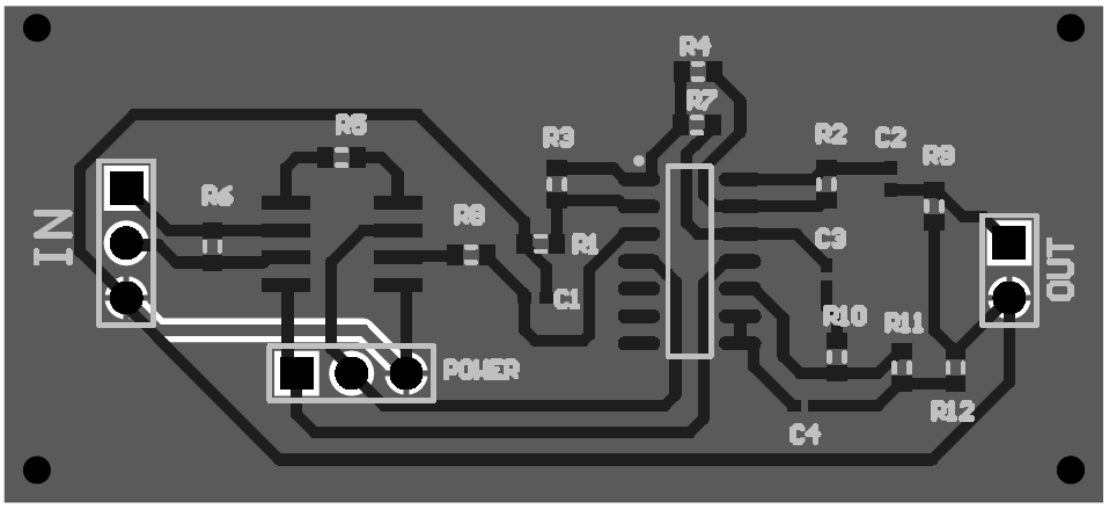


Figura 3.4 PCB sensor EMG con componentes SMD

Para el diseño de la placa PCB se tomó en cuenta elementos SMD como se observa en la figura 3.4, con el fin de que ocupe el menor espacio posible. Esta placa es de dimensiones de 2x5 cm y es para impresión en dos caras.

3.1.2 Análisis de costo del sensor

Comment	Description	Designator	Footprint	LibRef	Quantity	Unit Price	Total
OP484ES	Precision Rail-to-Rail Input and Output Operational Amplifier		SO-14_L	OP484ES	1	5	5
Cap Semi	Capacitor (Semiconductor SIM Model)	C1, C2, C3, C4	0402	Cap Semi	4	1	4
AD620	Instrumentation Amplifier 1 Circuit 8-SOIC	IC1	*SO8	AD620	1	2,15	2,15

Header 3	Header, 3-Pin	IN, POWER	HDR1X3	Header 3	2	0,3	0,6
Header 2	Header, 2-Pin	OUT	HDR1X2	Header 2	1	0,5	0,5
Res3	Resistor	R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12	J1-0603	Res3	12	0,12	1,44
Impresión					1	5	5
TOTAL							18,69

Tabla 3.1 Precio de los componentes del sensor

En la tabla 3.1 se desglosó el precio del sensor donde se tiene un valor económico en comparación a los sensores anteriormente mencionados.

3.1.3 Resultados de la Red Neuronal

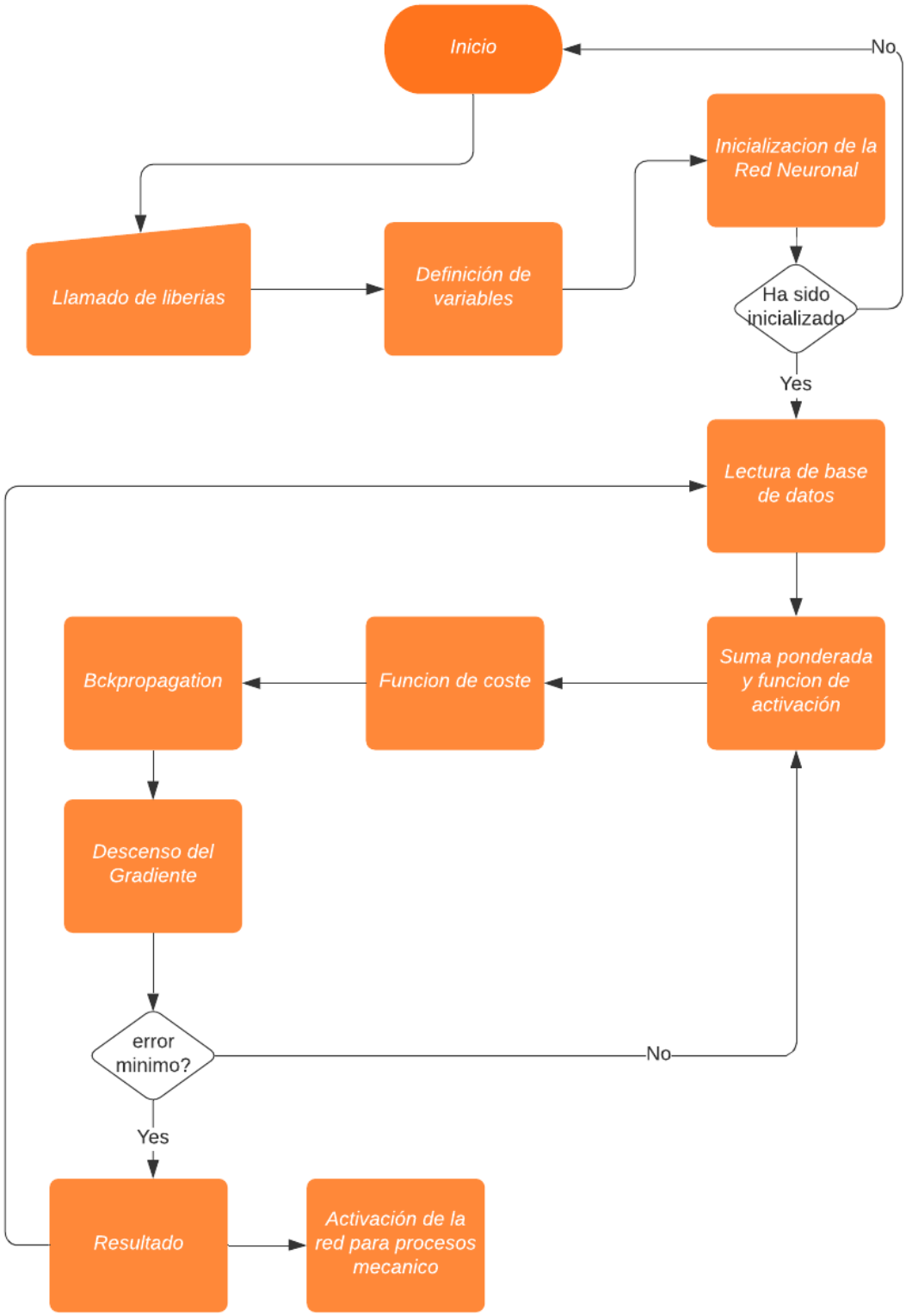


Figura 3.5 Comportamiento de la Red Neuronal durante el entrenamiento

En la figura 3.5 muestra el comportamiento de la red neuronal cuando está trabajando. A continuación, explicaremos el diagrama.

Al inicio del código llama a las librerías: numpy, matplotlib, pandas, seaborn, RPI.GPIO. Después se inicializaron todas las variables y fueron asignados los pines usados para la activación de los motores.

La Red Neuronal se activa con la función init, en la cual se inicializo los pesos y las funciones de activación. Posteriormente fueron leídas la base de datos donde se muestran las señales procesados por los sensores EMG.

Se continuo a leer la base de datos y se ejecutó la suma ponderada de las variables de entrada para luego alterarlas con las funciones de activación. Finalmente se obtiene la salida y se verifica la función de coste. Obtenida la función de coste se realiza Backpropagation para calcular las derivadas parciales, con estas últimas podremos obtener el descenso del gradiente. El descenso del gradiente optimiza la función de coste y de esta manera me permite entrenar a la red.

Si el error es mínimo se obtiene un resultado, este resultado es la orden de activación para los motores. Para el entrenamiento el número de iteraciones fueron de 15000 y se obtuvo un error del 4% es decir la eficiencia de la red neuronal es de 96%.

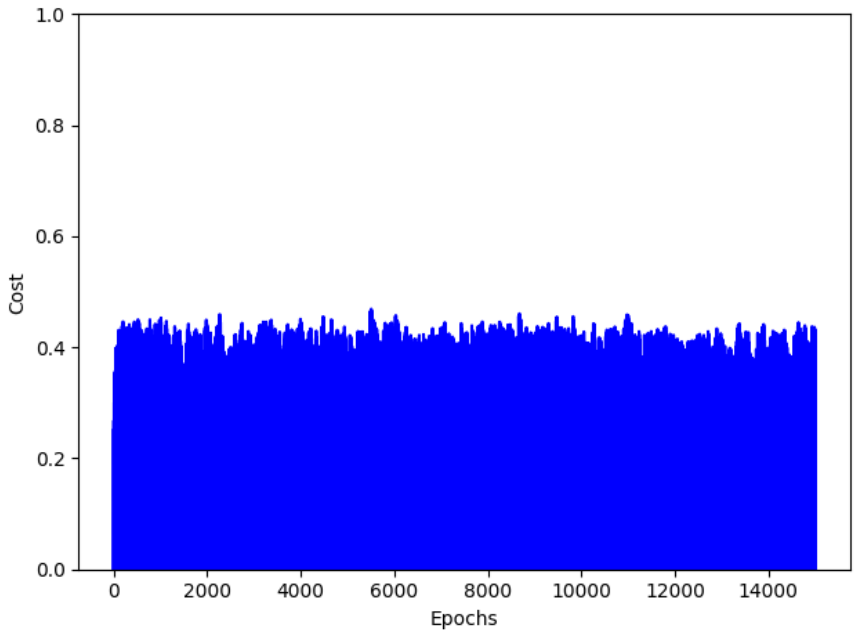


Figura 3.6 Grafico de resultados de Costo vs épocas

En la figura 3.4 se muestra la gráfica costo vs épocas la cual permite analizar el esfuerzo que produce el algoritmo al dispositivo cuando este se encuentra procesando los datos acordes un valor asignado de iteraciones dadas a la función epochs, es así que se te tiene un valor medio de un 0.4, mostrando que es un nivel aceptable ya que se maneja en rangos ascendente es decir su valor más alto es 1 que representa el máximo esfuerzo que se está generando y 0 como mínimo.

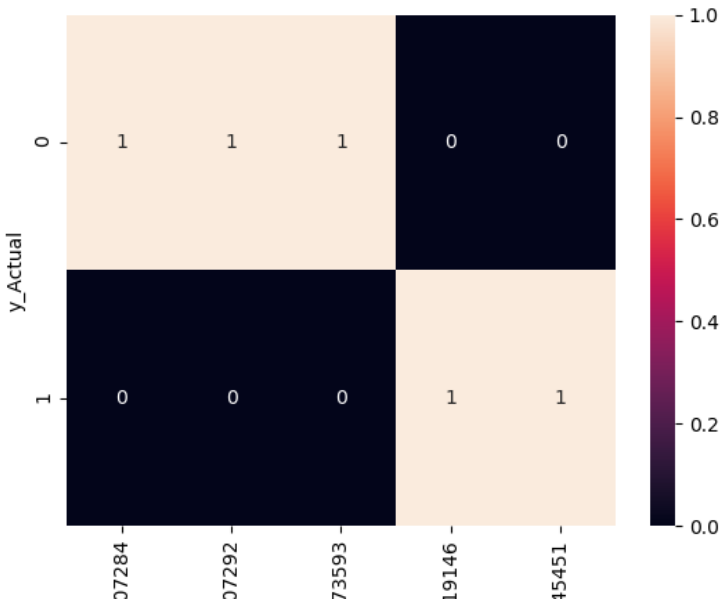


Figura 3.7 Matriz de Confusión

En la figura 3.7 se muestra la gráfica de la matriz de confusión la cual nos indica la precisión y la exactitud del modelo. Para el análisis toma valores actuales de [1 1 0 0 0] y los valores de predicción son tomados de la interacción cincuenta del modelo [0.94719146, 0.94745451, 0.00073593, 0.0007284, 0.0007292]. La figura previa nos muestra que si los valores de predicción son próximos a cero o a uno. Cuando el valor actual es 1 y el valor de predicción es próximo a uno entonces la matriz nos da un positivo verdadero, es decir un 1. La mismo aplica si el valor actual es cero y el valor de predicción es próximo a cero, su resultado es un negativo positivo, es decir un 1.

3.1.4 Análisis de costo del proyecto.

Producto	Cantidad	Precio
Kit de Raspberry pi 4	1	106.50
Motores lineales Dc Actuonix L12S	5	375.92
2 sensores EMG	2	37.40
ADS1115	1	5.00
Motor-Driver	3	12.00
cables	1	10.00
Batería 40000mA	1	63.00
Subtotal		609.82
Utilidades 30%		182.95
Total		792.77

Tabla 3.2 Valor del proyecto.

En la tabla 3.2 se muestra el gasto total del proyecto donde cabe recalcar que la mayoría de los componentes se pueden encontrar en el mercado nacional a excepción de los motores lineales que se los adquiere bajo pedido. Además, en la tabla que no se incluyó la estructura del exoesqueleto ya que se puede considerar que estaríamos delimitando las opciones para el uso del sistema, se debe tener en cuenta que este sistema está hecho para usarlo en cualquier tipo de exoesqueleto siempre y cuando funcione con señales electromiograficas.

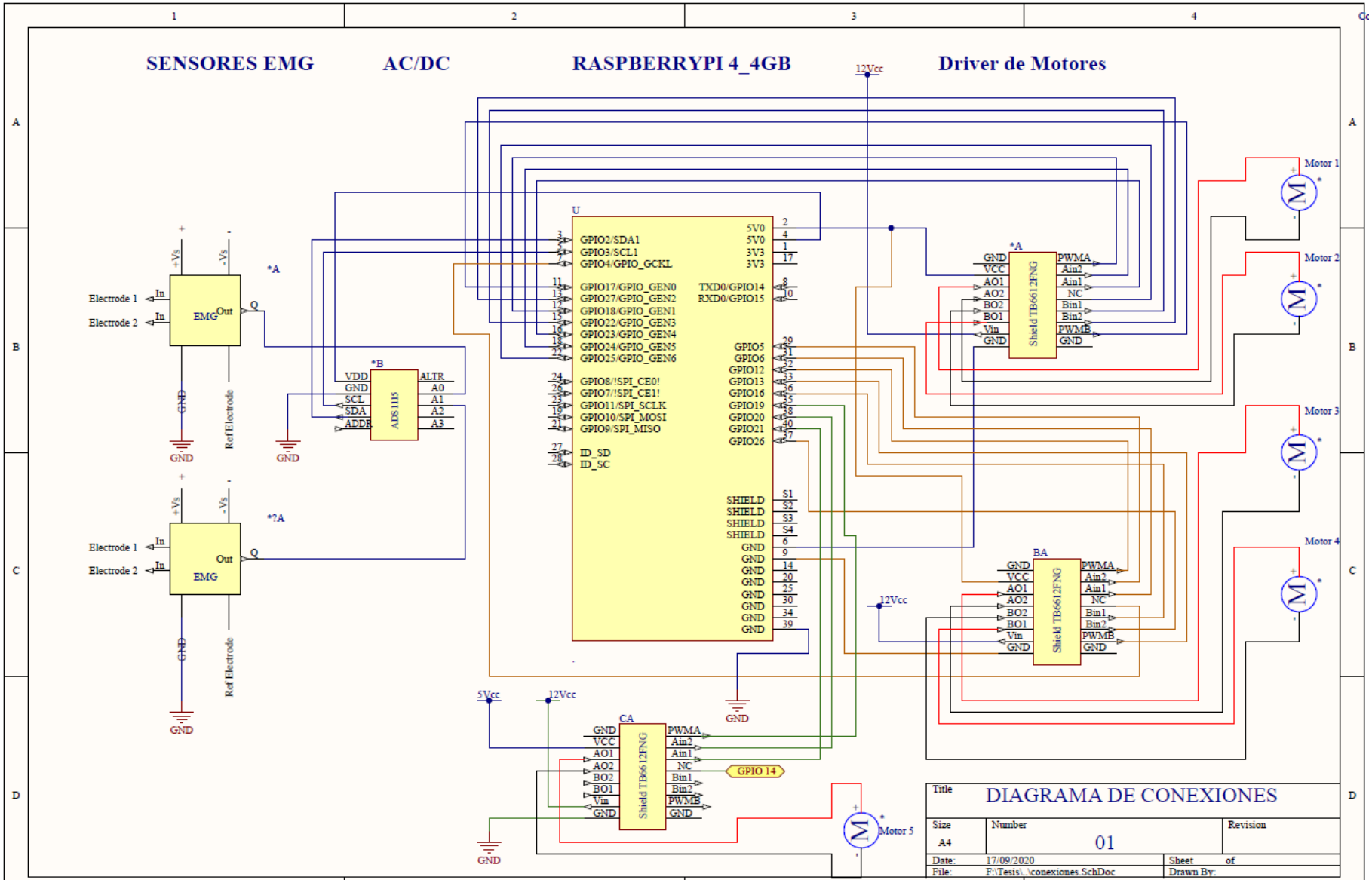


Figura 3.8 Esquemático de conexiones del sistema

En la figura 3.8 se muestra todas las conexiones del sistema, partiendo de dos sensores EMG. Fueron necesarios dos sensores debido a que los tendones de la mano tienden a ubicarse en determinadas zonas, tal cual se observa en la figura 3.8 y se explica detalladamente a continuación:

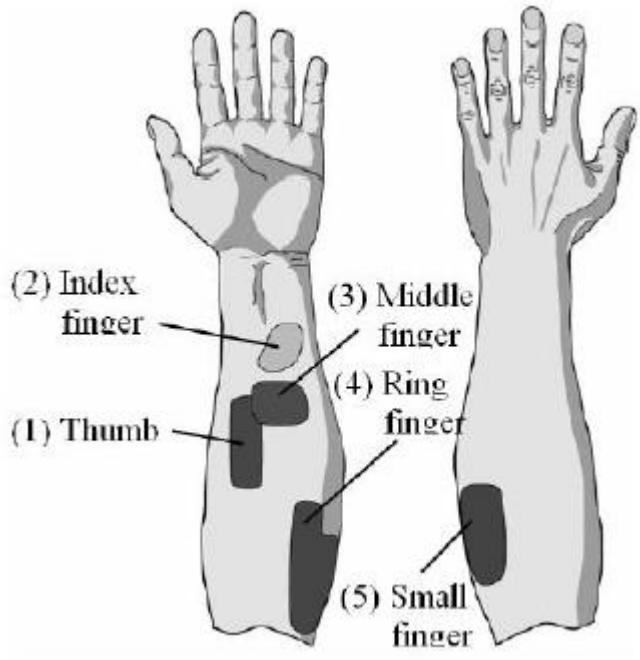


Figura 3.9 Ubicación de electrodos en zonas identificadas.

Al tener dos electrodos por sensor estos permiten captar señales en delimitadas zonas, como por ejemplo las zonas (1,2 y 3) de la figura 3.9 pueden ser cubiertas por un sensor. Pero queda libre las zonas (4 y 5) siendo necesario de otro sensor. Además, que se tiene un tercer electrodo, pero a este se lo toma como referencia estática por lo cual se lo ubica zonas de poca movilidad como pueden ser en la muñeca o codo.

Continuando con la explicación de la figura 3.8 el uso de los pines de la Raspberry pi 4 fueron acorde los requerimientos que amerita la programación siendo

los PWM los más usados y configurados para la comunicación con los controladores de motores y posteriormente a los motores.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Se diseño un sistema electrónico para exoesqueletos de mano que permite controlar los movimientos básicos tales como flexión y extensión de los dedos mediante captación de señales de electromiografía. Para lo cual se pudo analizar técnicamente los elementos que implican en este proyecto, además de mantener como objetivo que sean de bajo costo. Algunos de estos elementos por sus diseños complejos se debieron adquirirlos sin importar el precio como es la tarjeta embebida “Raspberry Pi 4” mientras otros elementos fueron necesarios diseñarlos como es el sensor EMG. El sensor EMG fue diseñado con un elemento de aplicaciones médicas como el amplificador de instrumentación medica AD620 y filtros tanto pasivos como activos de primer orden, al usar estas configuraciones donde no se necesita de muchos elementos se está evitando la acumulación de elementos y por ende que aparezcan ruidos externos a más de los 60Hz que se presenta en todos los componentes.

Una Raspberry Pi 4 no tiene entradas análogas por lo cual fue necesario incorporar el dispositivo ADS1115 este es un conversor análogo digital (ADC) de 16

bits donde se destaca por tener una gran precisión y lectura de valores negativos siendo ideal para el sistema. Para las salidas de la tarjeta fue necesario incorporar controladores debido a que son 5 motores los que se requirieron para mover cada dedo y la tarjeta Raspberry no cuenta con la suficiente corriente para activarlos a todos a la vez.

El consumo de energía dado en la tabla 2.6 se aproxima a los 6100mAh por lo cual para que tenga un tiempo aceptable de uso se colocó un powerbank de 40000mA, y con la ecuación 2.5 se pudo determinar que la duración de las baterías es de 6,55 horas, siendo relevante ya que si se compara con una sesión de rehabilitación estas duran 2 horas. Con el uso de estos exoesqueletos se espera una estimulación constante a los músculos afectados y que el tiempo de recuperación de las personas con ACV sean más cortas.

Al hacer uso de una base de datos de una publicación científica el proyecto tiende adaptarse a los parámetros con que estos fueron realizados y es donde obligo a de cierto modo a que el algoritmo sea más específico y entrenado para estos datos.

La red neuronal creada con propias funciones permitió tener un valor de error bajo siendo aproximado al 4% este se califica como un valor aceptable que a su vez que permite ver que el algoritmo no está especializado, es decir puede trabajar con nuevos datos y no solo con los que han sido usados para entrenar a la red neuronal. Además, de ser un algoritmo que genera un esfuerzo medio al procesador, permite

asegurar que puedan crearse a futuro nuevas mejoras e ir añadiendo más algoritmos que den más ejercicios de rehabilitación.

En la tabla 3.2 se muestra el costo total del proyecto donde se ha considerado los valores más bajos para su implementación y se espera que el prototipo ensamblado por completo no incremente su valor considerablemente ya que al comparar con exoesqueletos de alta gama su costo que se lo podría calificar como “sumamente barato”, el ahorro se encuentra aproximadamente con un 82% si se compara con un exoesqueleto standard de un hospital pero incrementaría el ahorro si se compara con exoesqueletos de más alto valor llegando a ser 10% o menor .

Recomendaciones

Al trabajar con la tarjeta embebida es necesario que esta tenga elementos de disipación de calor y carcasas que soporten altas temperaturas, ya que se pudo apreciar este inconveniente que a la larga puede causar quemaduras al paciente que se encuentre usando.

Al realizar el entrenamiento de la red neuronal se debe verificar que se procesen todos los datos, si por alguna razón no se cumplen se debe volver a entrenar, pero cambiando los pesos aleatorios iniciales. Es necesario indicar que si el entrenamiento de una red neuronal es lenta y tediosa cuando esta se haya completado no es necesario volver a realizarlo.

El error de una red neuronal no debe ser muy reducido al momento de ajustar sus pesos, ya que está causando que sea una red especializada es decir solo trabaje con los datos que han sido usados y que si llegan a ingresar nuevos datos esta generaría problemas.

Este proyecto ha sido desarrollado con miras a que en un futuro se profundicen en sus mejoras y se añadan nuevos movimientos, ya que el desarrollo de estos algoritmos conlleva un de tiempo extenso de análisis y se ha visto delimitado a su vez.

Al momento que se conecte todos los dispositivos como se muestra en la figura 3.4 estos deben tener una sola GND como referencia.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

2017, ©. C. (20 de 07 de 2020). Obtenido de El modelo de redes neuronales: www.ibm.com/support/knowledgecenter/es/ss3ra7_sub/modeler_mainhelp_client_ddita/components/neuralnet/neuralnet_model.html

Actuonix. (08 de 08 de 2020). *L12-R Small Linear Servo for RC and Arduino*. Obtenido de https://www.actuonix.com/L12-R-Linear-Servo-For-Radio-Control-p/l12-r.htm

Betz, M. R. (01 de 01 de 2011). *Accionamiento de motor*. Obtenido de [Figura]: Recuperado de https://www.sciencedirect.com/science/article/pii/B9780123820365000343?via%3Dihub

BitDegree. (21 de 01 de 2020). *División de conjuntos de datos con la función train_test_split de Sklearn*. Obtenido de https://www.bitdegree.org/learn/train-test-split

Clinic, M. (14 de 09 de 2019). *MayoClinic.org*. Obtenido de https://www.mayoclinic.org/es-es/diseases-conditions/stroke/in-depth/stroke-rehabilitation/art-20045172

Coras, C. A. (2012). *Procesamiento de señales de electromiografia superficial para deteccion de movimientos de dos dedos de la mano.* Lima: Universidad Ricardo Palma.

Cruz-Martínez, G. (01 de 03 de 2018). Diseño de Exoesqueleto con base en Cuatro Casos de Estudio de Rehabilitación de Miembro Superior. *Revista mexicana de ingeniería biomédica*, 39. Obtenido de http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0188-95322018000100081#B13

Device, A. (01 de 01 de 2011). *Instrumentation Amplifier.* Obtenido de [Figura]: Recuperado de https://www.analog.com/media/en/technical-documentation/data-sheets/AD620.pdf

EcuRed. (20 de 07 de 2020). *Baterias de Ion-Litio*. Obtenido de https://www.ecured.cu/Bater%C3%ADa_de_Ion_de_Litio

Esacademic. (01 de 01 de 2010). *Red neuronal artificial*. Obtenido de [Figura]: Recuperado de https://esacademic.com/dic.nsf/eswiki/994505

Figueroa, L. J. (12 de 09 de 2016). *Analisis de señales EEG para deteccion de eventos oculares, musculares y cognitivos.* Obtenido de Universidad Politecnica de Madrid: http://oa.upm.es/44379/1/TFM_LEONARDO_JOSE_GOMEZ_FIGUEROA.pdf

Hertfordshire, U. o. (05 de Marzo de 2015). *news.cision.com*. Obtenido de [Figura]: Recuperado de https://news.cision.com/university-of-hertfordshire/i/script,c1639144

interactivechaos. (19 de 08 de 2020). Obtenido de https://www.interactivechaos.com/manual/tutorial-de-machine-learning/escalado-de-datos

Kite. (20 de 08 de 2020). *Code Faster with Line-of-Code Completions, Cloudless Processing*. Obtenido de https://www.kite.com/python/docs/sklearn

Llamas, L. (14 de 11 de 2016). Obtenido de [Figura]: Recuperado de https://www.luisllamas.es/entrada-analogica-adc-de-16-bits-con-arduino-y-ads1115/

Llamas, L. (19 de 09 de 2017). *Ingenieria,informatica y diseño*. Obtenido de https://www.luisllamas.es/entrada-analogica-adc-de-16-bits-con-arduino-y-ads1115/#:~:text=El%20ADS1115%20es%20un%20conversor,6%20ADC%20de%2010%20bits.

Llamas, L. (s.f.). *ENTRADA ANALÓGICA DE 16 BITS CON ARDUINO Y ADC ADS1115*. Obtenido de [.

Morante, P. S. (01 de 11 de 2018). *Precauciones a la hora de normalizar datos en Data Science*. Obtenido de https://empresas.blogthinkbig.com/precauciones-la-hora-de-normalizar/

Mouser Electronics. (29 de 07 de 2020). Obtenido de https://www.mouser.ec/ProductDetail/584-AD620AN

naylamp. (05 de 09 de 2020). *naylampmechatronics*. Obtenido de [Figura]: Recuperado de https://naylampmechatronics.com/drivers/200-driver-puente-h-tb6612fng.html

Neurología, R. E. (07 de 05 de 2017). *revecuatneurol - Revista Ecuatoriana de Neurología*. Obtenido de

http://revecuatneurol.com/magazine_issue_article/enfermedad-cerebrovascular-ecuador-analisis-mortalidad-realidad-actual-recomendaciones/

Ozgur, C. (2017). MatLab contra Python contra R. *Diario Data Science*, 355-372.

Rami Khushaba, P. (04 de 08 de 2020). *Electromyogram (EMG) Repository*. Obtenido de https://www.rami-khushaba.com/electromyogram-emg-repository.html

Raspberry Pi 4. (01 de 03 de 2020). Obtenido de [Figura]: Recuperado de https://www.seeedstudio.com/Raspberry-Pi-4-Computer-Model-B-4GB-p-4077.html

Rosa, J. M. (12 de 08 de 2020). *Fundamentos Teoricos.* Obtenido de http://bibing.us.es/proyectos/abreproy/11375/fichero/MEMORIA%252FFundamentos+teoricos.pdf

Rubio, R. R. (1999). *Aplicaciones de las señales electromiográficas.* La Rioja: Dialnet.

sapiensman. (08 de 08 de 2020). *Servomotores lineales*. Obtenido de http://www.sapiensman.com/tecnoficio/electricidad/velocidad_de_motores_electricos4.php

Shenoi, B. (1965). Practical Realization of a Gyrator Circuit and RC-Gyrator Filters. *IEEE Transactions on Circuit Theory*, 374-380.

Shenzhen Nejifu Technology Co.,Ltd. (19 de 05 de 2015). Obtenido de [Figura]: Recuperado de https://nejifu.en.ec21.com/

Valencia, U. I. (21 de 03 de 2018). *Conceptos básicos de procesamiento de una señal digital.* Obtenido de https://www.universidadviu.com/conceptos-basicos-procesamiento-una-senal-digital/

Wiki, M. B.-S. (06 de 08 de 2020). *Wiki.seeedstudio*. Obtenido de https://wiki.seeedstudio.com/Motor_Bridge_Cape_v1.0/

APÉNDICES

APÉNDICE A

```
from time import sleep
import RPi.GPIO as GPIO
import numpy as np
from numpy import genfromtxt
import matplotlib.pyplot as plt
import pandas as pd
from pandas import read_csv, DataFrame, Series
import seaborn as sn
import time
import math

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False);

pwmFreq=100

#Setup Pins for motor controller
GPIO.setup(12,GPIO.OUT) #PWMA gpio 18
GPIO.setup(18,GPIO.OUT) #AIN2 gpio 24
GPIO.setup(16,GPIO.OUT) #AIN1 gpio 23  Thump
GPIO.setup(22,GPIO.OUT) #STBY gpio 25
GPIO.setup(15,GPIO.OUT) #BIN1 gpio 22  Index
```

```
GPIO.setup(13,GPIO.OUT) #BIN2 gpio 27
GPIO.setup(11,GPIO.OUT) #PWMB gpio 17

GPIO.setup(32,GPIO.OUT) #PWMC   gpio 12
GPIO.setup(29,GPIO.OUT) #AIN2.2 gpio 5
GPIO.setup(31,GPIO.OUT) #AIN1.2 gpio 6  Middle
GPIO.setup(7,GPIO.OUT) #STBY.2  gpio 4
GPIO.setup(36,GPIO.OUT) #BIN1.2 gpio 16 Ring
GPIO.setup(37,GPIO.OUT) #BIN2.2 gpio 26
GPIO.setup(33,GPIO.OUT) #PWMD   gpio 13

GPIO.setup(35,GPIO.OUT) #PWMD   gpio 19
GPIO.setup(38,GPIO.OUT) #AIN2.3 gpio 20
GPIO.setup(40,GPIO.OUT) #AIN1.3 gpio 21  Pinky
GPIO.setup(8,GPIO.OUT) #STBY.3  gpio 14

pwma=GPIO.PWM(12,pwmFreq)
pwmb=GPIO.PWM(11,pwmFreq)
pwmc=GPIO.PWM(32,pwmFreq)
pwmd=GPIO.PWM(33,pwmFreq)
pwme=GPIO.PWM(35,pwmFreq)

pwma.start(100)
pwmb.start(100)
pwmc.start(100)
```

```
pwmd.start(100)
pwme.start(100)

opcion=0



# # # # # # # # # # # # # # # # # # # # # CREACION DE LA RED NEURONAL
##################################################
def sigmoid(x):
    return 1.0/(1.0 + np.exp(-x))

def sigmoid_derivada(x):
    return sigmoid(x)*(1.0-sigmoid(x))

def tanh(x):
    return np.tanh(x)

def tanh_derivada(x):
    return 1.0 - x**2


class NeuralNetwork:

    def __init__(self, layers, activation='tanh'):
```

```
        if activation == 'sigmoid':
            self.activation = sigmoid
            self.activation_prime = sigmoid_derivada
        elif activation == 'tanh':
            self.activation = tanh
            self.activation_prime = tanh_derivada

        # inicializo los pesos
        self.weights = []
        self.deltas = []
        # capas = [2,3,2]
        # rando de pesos varia entre (-1,1)
        # asigno valores aleatorios a capa de entrada y capa oculta
        for i in range(1, len(layers) - 1):
            r = 2*np.random.random((layers[i-1] + 1, layers[i] + 1)) -1
            self.weights.append(r)
        # asigno aleatorios a capa de salida
        r = 2*np.random.random( (layers[i] + 1, layers[i+1])) - 1
        self.weights.append(r)

    def fit(self, X, y, learning_rate=0.2, epochs=100000):
        # Agrego columna de unos a las entradas X
        # Con esto agregamos la unidad de Bias a la capa de entrada
        ones = np.atleast_2d(np.ones(X.shape[0]))
        X = np.concatenate((ones.T, X), axis=1)
```

```
for k in range(epochs):
    i = np.random.randint(X.shape[0])
    a = [X[i]]

    for l in range(len(self.weights)):
            dot_value = np.dot(a[l], self.weights[l])
            activation = self.activation(dot_value)
            a.append(activation)
    # Calculo la diferencia en la capa de salida y el valor obtenido
    error = y[i] - a[-1]
    deltas = [error * self.activation_prime(a[-1])]

    # Empezamos en el segundo layer hasta el ultimo
    # (Una capa anterior a la de salida)
    for l in range(len(a) - 2, 0, -1):
        deltas.append(deltas[-1].dot(self.weights[l].T)*self.activation_prime(a[l]))
    self.deltas.append(deltas)

    # invertir
    # [level3(output)->level2(hidden)]  => [level2(hidden)->level3(output)]
    deltas.reverse()

    # backpropagation
    # 1. Multiplcar los delta de salida con las activaciones de entrada
```

```
        #    para obtener el gradiente del peso.
        # 2. actualizo el peso restandole un porcentaje del gradiente
        for i in range(len(self.weights)):
            layer = np.atleast_2d(a[i])
            delta = np.atleast_2d(deltas[i])
            self.weights[i] += learning_rate * layer.T.dot(delta)

        if k % 10000 == 0: print('epochs:', k)

    def predict(self, x):
        ones = np.atleast_2d(np.ones(x.shape[0]))
        a = np.concatenate((np.ones(1).T, np.array(x)), axis=0)
        for l in range(0, len(self.weights)):
            a = self.activation(np.dot(a, self.weights[l]))
        return a

    def print_weights(self):
        print("LISTADO PESOS DE CONEXIONES")
        for i in range(len(self.weights)):
            print(self.weights[i])

    def get_deltas(self):
        return self.deltas
```

```
# funcion para reconocer las entrada de señales y la activacion de motores


ti = NeuralNetwork([8,10,5],activation ='sigmoid')

hc = NeuralNetwork([8,6,5],activation ='tanh')


array1=np.genfromtxt(r'data1/HC_1.csv',delimiter=',',dtype=None)

array2=np.genfromtxt(r'data1/T_I1.csv',delimiter=',',dtype=None)

# array3=np.genfromtxt(r'data1/T_M1.csv',delimiter=',',dtype=None)

# array4=np.genfromtxt(r'data1/IMR1.csv',delimiter=',',dtype=None)

# array5=np.genfromtxt(r'data1/I_M1.csv',delimiter=',',dtype=None)


X1 = array1[1:6, 0:8] ## 5 filas y 8 columnas  HC

X2 = array2[10:15, 0:8] ## 5 filas y 8 columnas TI


# y = np.array([[1, 1, 1, 1, 1],    # mano cerrada H - C

#               [1, 1, 0, 0, 0],    # pulgar e indice T - I

#               [1, 0, 0, 1, 0],    # pulgar y anular T - M

#               [0, 1, 1, 1, 0],   #  levanta el meñique  I - M - R

#               [0, 0, 1, 1, 1]])  # formar el numero 3 con la mano I - M

#


y1 = np.array([[1, 1, 1, 1, 1],    # mano cerrada H - C

              [0, 0, 0, 0, 0],    #

              [0, 0, 0, 0, 0],    #

              [0, 0, 0, 0, 0],   #
```

```
            [0, 0, 0, 0, 0]])  #



y2 = np.array([[0, 0, 0, 0, 0],    #
            [1, 1, 0, 0, 0],    # pulgar e indice T - I
            [0, 0, 0, 0, 0],    #
            [0, 0, 0, 0, 0],   #
            [0, 0, 0, 0, 0]])  #



hc.fit(X1, y1, learning_rate=0.3, epochs=50000)
ti.fit(X2, y2, learning_rate=0.3, epochs=50000)




index2=0
serie2=[]
for e in X2:

    print("X2:",e,"y2:",y2[index2],"Network:",ti.predict(e))
    serie2= ti.predict(e)
    np.append(ti, ti.predict(e), axis= None) ###arreglo del comportamiento T-I
    index2=index2+1
```

```
print("\n \n serie 2 \n", serie2)
##########################################################################
#################
index1=0
serie1=[]
for i in X1:

    print("X1:",i,"y1:",y1[index1],"Network:",hc.predict(i))
    serie1= hc.predict(i)
    np.append(hc, hc.predict(i), axis= None) ###arreglo del comportamiento H-C
    index1=index1+1



print("\n \n serie 1 \n", serie1)
##########################################################################
###################

deltas2 = ti.get_deltas()
valores2=[]
index2=0

for arreglo in deltas2:
    valores2.append(arreglo[1][0] + arreglo[1][1])
    index2=index2+1
```

```
##########################################################################
##################
deltas1 = hc.get_deltas()
valores1=[]
index1=0

for arreglo in deltas1:
    valores1.append(arreglo[1][0] + arreglo[1][1])
    index1=index1+1

# plt.plot(range(len(valores2)), valores2, color='b')
# plt.ylim([0, 0.5])
# plt.ylabel('Cost')
# plt.xlabel('Epochs')
# plt.tight_layout()
# plt.show()

df_ti= pd.DataFrame(serie2)
df_hc= pd.DataFrame(serie1)

print("\n columna HC\n", df_hc)
print("\n columna TI\n", df_ti)
print("\n Matriz ")

# df= pd.read_csv('entrenamiento.csv')
```

```
# df = pd.DataFrame(df_hc, columns=['HC'])
#
# df.loc[:0, 'HC'] = df_hc  ### agrego una primera columna
# df.loc[:1, 'TI'] = df_ti  ### agrego una segunda columna

df= pd.merge(df_hc, df_ti, right_index= True, left_index= True)

df.to_csv('entrenamiento.csv',header=None)
print(df)

# # # # # # # # # # # # # # # matriz de confusion#####################################################################
#####################################

data = {'y_Actual':    [1, 1, 0, 0, 0], # se hizo una captura en el nivel 50
        'y_Predicted': [0.94719146, 0.94745451, 0.00073593, 0.0007284,  0.0007292 ]
        }

df_data = pd.DataFrame(data, columns=['y_Actual','y_Predicted'])
confusion_matrix = pd.crosstab(df_data['y_Actual'], df_data['y_Predicted'] ) #,
rownames=['Actual'], colnames=['Predicted'], margins = True

sn.heatmap(confusion_matrix, annot=True)

plt.show()
```

```
# # # # # # # # # # # # # # # FIN matriz de confusion#########################################################################################################################################
```

```
sumafirst= df['0_x'].sum() # hc
sumasecond= df['0_y'].sum() # ti

print("\n suma HC\n", sumafirst)
print("\n suma TI\n", sumasecond)

#if (sumafirst > 3):
#    print("cerrar mano")
#    opcion=2
#    movimiento_motor(opcion)
#elif (sumasecond > 1.5):
#    print("\nunir dedo pulgar e indice\n")
#    opcion=3
#    movimiento_motor(opcion)
#else:
#    print("\nquedar en la posicion original\n ")
#    opcion=1
```

```
#    movimiento_motor(opcion)




#
##########################################################################funcion
motor################################
############################################################

#Functions para habilitar motores
############################################################
def OpenFingers(spd):   #Abre los dedos
    runMotor(0, spd, 1)
    runMotor(1, spd, 1)
    runMotor(2, spd, 1)
    runMotor(3, spd, 1)
    runMotor(4, spd, 1)



def CloseFingers(spd): #Cierra los dedos
    runMotor(0, spd, 0)
    runMotor(1, spd, 0)
    runMotor(2, spd, 0)
    runMotor(3, spd, 0)
```

```
    runMotor(4, spd, 0)

def Thump(spd):        # Activa solo el dedo pulgar
    runMotor(0, spd, 1)
    runMotor(1, spd, 0)
    runMotor(2, spd, 0)
    runMotor(3, spd, 0)
    runMotor(4, spd, 0)

def Index(spd):        # Activa solo el dedo Indice
    runMotor(0, spd, 0)
    runMotor(1, spd, 1)
    runMotor(2, spd, 0)
    runMotor(3, spd, 0)
    runMotor(4, spd, 0)

def Middle(spd):       # Activa solo el dedo medio
    runMotor(0, spd, 0)
    runMotor(1, spd, 0)
    runMotor(2, spd, 1)
    runMotor(3, spd, 0)
    runMotor(4, spd, 0)

def Ring(spd):         # # Activa solo el dedo anular
    runMotor(0, spd, 0)
```

```
    runMotor(1, spd, 0)
    runMotor(2, spd, 0)
    runMotor(3, spd, 1)
    runMotor(4, spd, 0)

def Pinky(spd):        ## Activa solo el dedo meñique
    runMotor(0, spd, 0)
    runMotor(1, spd, 0)
    runMotor(2, spd, 0)
    runMotor(3, spd, 0)
    runMotor(4, spd, 1)

#Funcion para activar
def runMotor(motor,spd,direction):
    GPIO.output(8,GPIO.HIGH)
    GPIO.output(7,GPIO.HIGH)
    GPIO.output(22,GPIO.HIGH)
    in1=GPIO.HIGH
    in2=GPIO.LOW

    if(direction==1):
        in1=GPIO.LOW
        in2=GPIO.HIGH

    if(motor==0):
```

```
    GPIO.output(16,in1)
    GPIO.output(18,in2)
    pwma.ChangeDutyCycle(spd)

elif(motor==1):
    GPIO.output(15,in1)
    GPIO.output(13,in2)
    pwmb.ChangeDutyCycle(spd)

elif(motor==2):
    GPIO.output(31,in1)
    GPIO.output(29,in2)
    pwmc.ChangeDutyCycle(spd)

elif(motor==3):
    GPIO.output(36,in1)
    GPIO.output(37,in2)
    pwmd.ChangeDutyCycle(spd)

elif(motor==4):
    GPIO.output(40,in1)
    GPIO.output(38,in2)
    pwme.ChangeDutyCycle(spd)
```

```
def motorStop():
    GPIO.output(8,GPIO.LOW)
    GPIO.output(7,GPIO.LOW)
    GPIO.output(22,GPIO.LOW)


#Main
################################################
def main(opcion):
    while True:


        if (sumafirst>3):
            CloseFingers(100) #Run motor reverse
            sleep(2)    #... for 2 seconds
            motorStop() #...stop motor
            sleep(.250) #deley between motor runs


        elif sumasecond > 1.5:      #activo el pulgar y indice
            Thump(100) #Run motor 0
            Pinky(100) #Run motor 4
            sleep(2)    #... for 2 seconds
            motorStop() #...stop motor
            sleep(.250) #deley between motor runs
```

```
        else:
            OpenFingers(100) #Run motor forward
            sleep(2)    #... for 2 seconds
            motorStop() #...stop motor
            sleep(.250) #deley between motor runs

#         elif sumafirst==1001:#'index':
#             Index(100) #Run motor 1
#             sleep(2)    #... for 2 seconds
#             motorStop() #...stop motor
#             sleep(.250) #deley between motor runs
#
#         elif sumafirst==1002:#'middle':
#             Middle(100) #Run motor 2
#             sleep(2)    #... for 2 seconds
#             motorStop() #...stop motor
#             sleep(.250) #deley between motor runs
#
#         elif sumafirst==1003:#'ring':
#             Ring(100) #Run motor 3
#             sleep(2)    #... for 2 seconds
#             motorStop() #...stop motor
#             sleep(.250) #deley between motor runs
#
# #         elif sumafirst<1004:#'pinky':
```

```
# #           Pinky(100) #Run motor 4
#             sleep(2)    #... for 2 seconds
#             motorStop() #...stop motor
#             sleep(.250) #deley between motor runs



if __name__=="__main__":
    main(opcion)
```