Universidad Católica Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería Mecánica, Mecánica-Eléctrica y Mecatrónica

CONTROL POR OPTOMIOGRAFÍA (OMG) Y CONTRUCCIÓN DE UNA MANO ROBÓTICA CON TECNOLOGÍA DE IMPRESIÓN 3D

Tesis presentada por el Bachiller:

Zegarra Chaguayo, Eduardo David

Para optar por el Título Profesional de

Ingeniero Mecatrónico

Asesor:

Ing. Mestas Ramos, Sergio

Arequipa – Perú

2019

Universidad Católica de Santa María

(51 54) 382038 Fax:(51 54) 251213 ucsm@ucsm.edu.pe http://www.ucsm.edu.pe Apartado:1350

AREQUIPA - PERÚ

ESCUELA PROFESIONAL DE INGENIERÍA MECANICA, MECÁNICA ELÉCTRICA Y MECATRÓNICA

INFORME DICTAMINATORIO

V I S T O

EL BORRADOR DE TESIS TITULADO:

"CONTROL POR OPTOMIOGRAFIA (OMG) Y CONSTRUCCION DE UNA MANO ROBOTICA CON TECNOLOGIA DE IMPRESIÓN 3D"

Presentado por el Bachiller:

ZEGARRA CHAGUAYO EDUARDO DAVID

Nuestro **DICTAMEN** es:

Favorable

OBSERVACIONES:

Ninguna

Arequipa, 08 de marzo del 2019

ING. SERGIO MESTAS RAMOS

ING. JUAN CARLOS CUADROS MACHUCA

AGRADECIMIENTOS

A mis padres Tomas y Elsa por sus sacrificios, apoyo constante, sabios consejos, cariño y amor, los Amo.

A la vida que tanto me ha enseñado con su famoso *"NO GAME, NO LIFE"* que significa: no hay vida sin juego, cada vez que avanzamos subimos a otro nivel y los retos se equilibran a ese nivel, lo que es un constante avance.

A Rodrigo, a toda mi familia que nunca dejo de creer mí y siempre me dio su apoyo incondicional.

A Dios por darme sabiduría y perseverancia.

Gracias a mi esfuerzo y apoyo de ellos, para cumplir mis metas.

EDUARDO DAVID ZEGARRA CHAGUAYO.

INDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Para el desarrollo del presente proyecto se tiene como base el control y la implementación del hardware en una mano robótica con tecnología de impresión 3D.

Cumpliendo con el objetivo principal de la tesis es construir una mano robótica con tecnología de impresión 3D e implementar control por optomiografía y entonces lograr emular los movimientos de extensión y flexión de los dedos con la adquisición realizada por sensores fotoeléctricos en la piel.

Con la implementación del control por optomiografía en el microcontrolador se podrá desarrollar el desplazamiento de los dedos de la mano robótica, la configuración de los parámetros a controlar y lograr realizar el monitoreo de todos los parámetros de entrada y salida.

El presente documento consta de cuatro capítulos además de conclusiones, observaciones, bibliografía y los anexos.

En el capítulo uno se explicará los mecanismos utilizados para el análisis de la problemática de investigación y así poder ejecutar el proyecto.

En el capítulo dos se explican las nociones y criterios básicos teóricos para poder desarrollar el proyecto.

En el capítulo tres, se realiza el desarrollo de la ingeniería como el diseño y selección mecánica, electrónico y la programación del proyecto.

Finalmente, en el capítulo cuatro, se mostrará la implementación del proyecto finalizado e implementado para el desarrollo de las pruebas y corroborar el cumplimiento con los objetivos planteados para el desarrollo de los mismos.

Palabras claves:

Optomigrafía, Impresión 3D, Control, Mano robótica, Maquina de estados finitos.

ABSTRACT

For the development of this project is based on the control and implementation of the hardware in a robotic hand with 3D printing technology.

Fulfilling the main objective of the thesis is to build a robotic hand with 3D printing technology and implement control by optomiography and then emulate the extension and bending movements of the fingers with the accomplished acquisition by photoelectric sensors on the skin.

With the implementation of the optomyography control in the microcontroller it will be possible to develop the displacement of the fingers of the robotic hand, the configuration of the parameters to be controlled and achieve the monitoring of all the input and output parameters.

This document consists of four chapters as well as conclusions, observations, bibliography and annexes.

Chapter one will explain the mechanisms used for the analysis of the research problem so that the project can be implemented.

Chapter two explains the basic notions and theoretical criteria for developing the project.

In chapter three, the development of engineering is carried out, such as design and mechanical and electronic selection and project programming.

Finally, in chapter four, it will show the implementation of the completed and implemented project for the development of the tests and verify compliance with the objectives set for the development of the tests.

Key words:

Optomyography, 3D printing, Control, Robotic hand, Finite state machine.

CAPITULO I
MARCO METODOLÓGICO

CAPITULO I

MARCO METODOLÓGICO

1.1. IDENTIFICACIÓN Y DESCRIPCIÓN DEL PROBLEMA.

La creciente necesidad del uso de las prótesis tiene como base diversos orígenes, que se encuentran clasificadas en las amputaciones traumáticas, amputaciones por cáncer, incidencias por causas congénitas y las amputaciones cardiovasculares siendo estas la más impactantes, una de las enfermedades es la diabetes que es una de las principales razones de amputaciones de las extremidades superiores en cada parte del mundo y todos los años. Sabemos que nuestras manos son muy importantes para poder desarrollar diversas actividades en nuestro día a día.

Actualmente existen diferentes tipos de prótesis de la mano, algunas pudiendo emular el comportamiento de la mano humana y otros solo prótesis estéticas a las cuales solo se puede acceder a costos muy altos, dando al usuario final la búsqueda de otras alternativas que puedan lograr reducir sus costos, para lo cual en la actualidad hay una nueva perspectiva en esta búsqueda de tecnologías como es el uso de la impresora 3D, con esta herramienta se podrá desarrollar la reconstrucción y la fabricación de este tipo de prótesis a un menor costo y dar mayor accesibilidad a las personas con este tipo de falencias y así poder reducir su costo de obtención al usuario final.

Las prótesis actuales que son producidas por grandes empresas estas tienen muchos grados de libertad en su funcionabilidad, pero estas suelen agotar al usuario final debido la presencia de gran volumen y peso, además de tener como desventaja los costos elevados, gran rigidez y además de poseer un rango limitado de movimiento.

Para poder controlar las prótesis se requiere adquirir la información del desplazamiento muscular para luego ser procesado con algún método de adquisición y control (según será requerido), el más común a utilizar es la electromiografía y con el uso de mío-sensores que adquieren la lectura de pulsos eléctricos y procesando esta información da opción al control, pero estos sistemas son costosos y presentan una mayor complejidad para adquirir más de una lectura, al aumentar los grados de libertad hace que el costo aumente. Los mío-sensores trabajan básicamente con electrodos artificiales invasivos o no invasivos.

Lograr controlar las prótesis con miografía no es una tarea simple debido que para poder controlar un grado de libertad como el de abrir y cerrar la mano requiere una serie de procesos y acondicionamientos para poder realizar esta acción, desde el punto de vista económico este tipo de sistema de acondicionamiento le da un valor añadido a la prótesis, logrando incrementar el costo, dando lugar a la búsqueda de alternativas que den una mayor factibilidad económica para poder implementarse y poder abaratar los costos de producción.

En la actualidad el impacto e influencia que está dando el usar las tecnologías que trabajen en 3D en los diversos campos al lograr dar una reducción significativa de precios de producción como por ejemplo en implantes como en prótesis, con esto se puede obtener mayor accesibilidad para muchas más personas además de darnos unas ventajas mecánicas como el bajo peso, robustez. Con el uso de esta tecnología en el campo de investigación y desarrollo, da la opción a la reducción de costos los sistemas mecatrónicos en nuestro enfoque y otros diferentes.

Utilizando estas ventajas y sabiendo que es necesario implementar como desarrollar un sistema de control que reducan el costo en el prototipo y en la

aplicación al usuario final para el uso de la prótesis y así poder ayudar al dar una mejor calidad de vida de esta persona.

1.1.1. CAMPO, ÁREA Y LÍNEA DE INVESTIGACIÓN.

Campo : Ingeniería

Área : Ingeniería Mecatrónica.

Línea : Robótica y Automatización

1.2. ANTECEDENTES

Alrededor del mundo existen diversas compañías e instituciones como también centros educativos que han realizado esfuerzos y puesto recursos en el diseño de sistemas que imiten en forma funcional y estructural a la mano humana. De la misma forma, en los últimos años se han logrado significativos avances en el desarrollo de sistemas de rehabilitación y de prótesis mecatrónicas dirigidas a personas que han perdido una extremidad. A continuación, se presenta algunos ejemplos del desarrollo sobresaliente con el uso estos recursos animatrónicos, manos y prótesis.

El sistema BEBIONIC [5] es una prótesis comercial articulada con sensores mioeléctricos (figura 1.1) que contiene microcontroladores que monitorean constantemente la posición de los dedos. Ha sido diseñada con un ensamble robusto que le proporciona gran resistencia al impacto. Tiene capacidad de configurar su velocidad y fuerza de agarre de manera inalámbrica y su máxima fuerza aplicable es de 75 N, en su sistema de alimentación emplea baterías de litio y puede programarse para tener 14 patrones de posición y agarre, posee un peso aproximado de 520 g. Además, posee motores de alta tecnología, ubicados en la palma de la mano que son empleados para el movimiento de cada dedo.

Figura 1.1: Mano comercial Bionic

Fuente: http://tiendaonline.juanbravo.com, mano biónica bebionic

La mano DEXTRUS [4] (figura 1.2) es una mano robótica no comercial que ofrece gran parte de la funcionalidad de una mano humana. En su estructura utiliza motores eléctricos en lugar de los músculos y los cables de acero en lugar de los tendones. Lo implementaron con tecnología de impresión 3D.

La mano se puede conectar a una prótesis existente utilizando un conector estándar. Utiliza electrodos revestidos para leer las señales de sus músculos restantes, que puede controlar la mano, para las condiciones de abrir y cerrar.

Figura 1.2: Mano robótica DEXTRUS

Fuente: https://bluebadgestyle.com

La mano INMOOV es un diseño que fue desarrollado por Gael Langevin para un proyecto propio que es la construcción de robots animatrónicos con piezas y partes impresas con una impresora 3D, siento está apoyada por la licencia de código abierto dando la facilidad a otras personas que puedan desarrollar otras aplicaciones con su diseño, entonces dado la similitud a las partes humanas estas fueron tomadas para realizar una prótesis artesanal, este proyecto fue iniciado en el 2012. En la aplicación de prótesis esta usa sensores miográficos cuyo funcionamiento básico es abrir y cerrar. En este sistema se usa el sensor el

"THALMIC"[1] para luego procesarse por electromiografía que no presenta costos bajos en las aplicaciones y otros módulos EMG en el desarrollo del mismo.

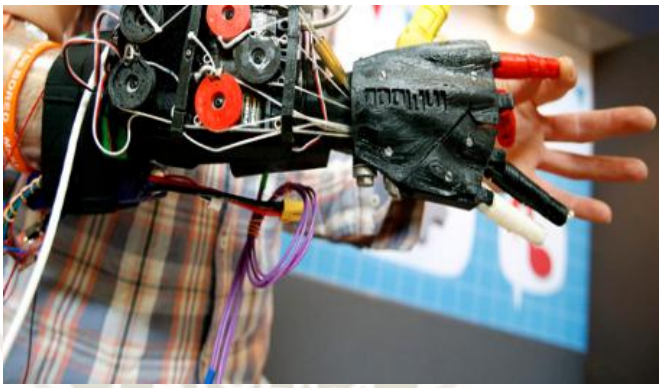


Figura 1.3: Control EMG INMOOV HAND

Fuente: https://nationalpost.com, Reportaje de partes biónicas

FINCH, prótesis de mano eléctrica de 3 dedos oponibles, de forma similar a la mano, es el brazo artificial eléctrica que la función es excelente como una herramienta [9].

El contacto humano incluso cuando los vasos en el hogar, se debe decir que incluso puede distinguirse de ser prótesis de mano.

Figura 1.4: Mano FINCH

Fuente: Laboratorio de prototipado y diseño de la universidad de Tokio

[1] Banda de brazo de gestos mioeléctricos desarrollado por la compañía Thalmic lab

Presenta buen funcionamiento para la vida cotidiana en comparación de las prótesis de mano estéticas, se supone que se utilizarán para trabajar en casa o en el trabajo. Con su control de 3 dedos oponibles, permite el funcionamiento con diversos objetos.

Presenta una función de sujeción del objeto de hasta 500 g, 350 g de peso total (incluyendo un socket), teniendo un tiempo continuo de funcionamiento: cerca de 7 horas, también posee un sistema operativo basado en la detección de los músculos planteada por un uso innovador de los sensores fotoeléctricos y otro beneficio adicional es poder fabricarlo mediante el uso de una impresora 3D.

Teniendo como antecedente de manos robóticas desarrollada en la universidad católica Santa María, encontramos la tesis de "Diseño y construcción de una mano protésica" elaborada en el año 2015, el cual se enfoca en el desarrollo mecánico y un sistema de abrir-cerrar accionada por señales entregadas por pulsadores logrando que los motores DC se desplacen en sentido horario o antihorario según el flujo de la corriente [10].

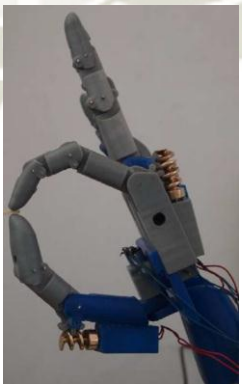


Figura 1.5: Diseño y construcción de una mano protésica

Fuente: Proyecto de titulación UCSM, 2015

1.3. OBJETIVOS

1.3.1. OBJETIVO PRINCIPAL

Construir una mano robótica con tecnología de impresión 3D e implementar control por optomiografía.

1.3.2. OBJETIVOS ESPECÍFICOS

- Construir una mano antropomórfica que sea robusta, segura, ligera y transportable.
- Diseñar el control de los actuadores de la mano robótica por optomiografía (OMG)
- Desarrollar una interfaz gráfica que facilite la calibración y visualización de los parámetros de control.
- Construir una banda para el antebrazo usando sensores fotoeléctricos que puedan detectar el cambio muscular
- Validar la operación del efector.

1.4. HIPOTESIS

Al implementar el sistema de control por OPTOMIOGRAFIA (OMG) logaremos controlar una mano robótica con diferentes grados de libertad desplazando el efector final además de reducir los costos de producción como poder demostrar la viabilidad funcional del proyecto.

1.5. VARIABLES

1.5.1. Variable dependiente

Controlar el desplazamiento del efector final de la mano robótica.

1.5.2. Variable independiente

Adquisición por un sensor IR a un microcontrolador, donde desarrollaremos el control por Optomiografía.

1.6. JUSTIFICACIÓN DE LA INVESTIGACIÓN

En la actualidad el desarrollo del sistema de control de prótesis está siendo objeto de estudio en diversas partes del mundo debido a que este tipo de control es complejo, avanzado y sobre todo presenta un gran desarrollo en el campo de la electromiografía.

Para realizar la investigación y el desarrollo del proyecto, se toma como referencia la creación de un sistema de control por optomiografía como una alternativa emergente e innovadora al utilizar sensores infrarrojos en la adquisición, teniendo como ventajas de este sensor como la velocidad de respuesta y nos presentan la facilidad de poder leer los presentes cambios en la contracción y extensión de los músculos de la mano al convertir estos a señales de voltaje finalmente poder desarrollar una lógica de control para que realice el desplazamiento en los actuadores finales y lograr reducir la complejidad del control, costos de producción y demostrar viabilidad funcional.

1.7. ALCANCES Y LIMITACIONES

El tema de estudio de la presente investigación, es el desarrollo del control por optomiografía de una mano robótica poder realizar la adquisición con el uso sensores fotoeléctricos que recibirán la información de movimientos de los músculos y convertirlos en voltaje, con esta data realizar la lógica de control en el microcontrolador seleccionado además de controlar los actuadores y poder realizar el desplazamiento de cada como poder demostrar el uso un sensor fotoeléctrico en el sistema de adquisición de las prótesis, con la optomiografía poder controlarlo y también de poder reducir los costos.

Para el sistema mecánico de la mano se dará uso de la tecnología de impresión 3D para el desarrollo del proyecto teniendo como consecuencia la disminución del coste de maquinado en el prototipo y en el proyecto final, además que el diseño será seleccionado para poder demostrar el funcionamiento y cumplimiento de la lógica de control desarrollada.

El método de adquisición de datos con el uso de sensor IR es innovador dando un gran aporte aplicativo, ya que ayudara a cuantificar las muestras adquiridas del cuerpo de la persona, también dando un gran beneficio como mayor accesibilidad a personas con bajos recursos económicos.

El tiempo de vida útil del sistema mecatrónico estará limitado por la capacidad de la fuente de alimentación y el torque máximo por los servomotores.

La distancia máxima del sensor no debe exceder el rango de 0.6 cm a 1 cm. Si supera el rango de lectura de los límites fijados como máximos y mínimos, tendrá como consecuencia que durante el proceso será influenciado por el ruido y se modificará la señal original.

1.8. ESTRUCTURA DEL TRABAJO

El trabajo presenta una división de **cuatro** capítulos principales, además de poseer conclusiones, bibliografía y anexos con información complementaria del proyecto. En el **capítulo uno** que es el marco metodológico donde explica la viabilidad y el porqué del desarrollo del proyecto, en el **capítulo dos** se desarrollará el marco teórico en el cual tendremos las nociones teóricas básicas para el entendimiento del proyecto. En el **capítulo tres** se realiza el desarrollo de la ingeniería donde tendremos el diseño, la selección, el desarrollo de programación y clasificación para poder desarrollar el proyecto. Por último, en el **capítulo cuatro** se realizarán las pruebas de funcionamiento del proyecto, pruebas de monitoreo y el ensamblaje general.

CAPITULO II
MARCO TEÓRICO

CAPITULO II

MARCO TEÓRICO

2.1. ROBÓTICA

Desde tiempos muy remotas el hombre ha deseado desarrollar máquinas que obtengan forma de seres humanos y le ayuden a realizar las operaciones que no le gustan, las que le resultan aburridas o peligrosas. A diferencia de una persona humana, una máquina nunca se cansaría ni se enfermaría y siempre estaría dispuesta a trabajar. Los elementos que pueden funcionar automáticamente se utilizan desde épocas tan remotas como la antigua Grecia, sin embargo, es hasta mediados del siglo veinte cuando se lograron materializar los primeros robots industriales. Estos robots industriales distaban mucho de los sueños de poder contar con una máquina con forma de ser humano. Casi cincuenta años después de la aparición de los primeros robots se sigue trabajando en el diseño y fabricación de estas máquinas similares al ser humano.

2.2. HISTORIA DE LAS PRÓTESIS[2]

En el camino del diseño de prótesis están se han relacionado directamente con el manejo de materiales que son empleados por el hombre, siendo necesario entender la biomecánica del cuerpo, así como el desarrollo tecnológico necesario. Las prótesis se desarrollaron con el fin de mejorar y reemplazar una función, un miembro del cuerpo humano afectado.

Antes del inicio de la era cristiana, los egipcios ya sustituían las manos por diferentes artefactos donde aquí se encuentra la primera prótesis de miembro superior registrada data del año 2000 a. C., fue encontrada en una momia egipcia;

[2] Bruno Sospedra Griño, 2015. Diseño mecánico de prótesis de mano multidedo antropomórfica infractuada, p.35

la prótesis estaba sujeta al antebrazo por medio de un cartucho adaptado al mismo.

Figura 2.1: Mano de Alt-Ruppin

Fuente: http://tiendaonline.juanbravo.com/

Muchos son los personajes que buscaron imitar la función de la mano tal es el caso que se desarrolló en el año de 1400 donde se fabricó la mano de Alt-Ruppin (Figura 2.1) construido hierro, constaba de un pulgar rígido en oposición y dedos flexibles, los cuales eran flexionados pasivamente.

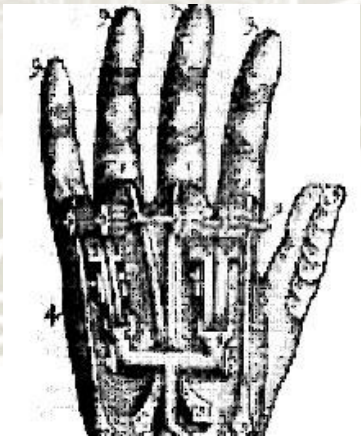


Figura 2.2: Primer brazo artificial móvil

Fuente: https://wikipedia.org

Es hasta el siglo XVI, que el diseño del mecanismo de las prótesis de miembro superior se ve mejorado considerablemente, gracias al médico militar francés Ambroise Paré, quien desarrolló el primer brazo artificial móvil al nivel

de codo, llamado Le petit Loraine el mecanismo era relativamente sencillo tomando en cuenta la época, los dedos podían abrirse o cerrarse presionando, también traccionando, fue la primera mano estética de cuero como prótesis de miembro superior se puede visualizar en la figura 2.2.

En el siglo XIX, entre las innovaciones más importantes al diseño de las prótesis de miembro superior, se encuentra la del alemán Peter Beil fue quien desarrolló una prótesis que permitía cerrar los dedos de una forma diferente usando los movimientos de tronco y hombros iniciándose así la etapa de las prótesis (Figura 2.3).

Figura 2.3: Prótesis de mano con pulgar móvil y gancho dividido sagitalmente

Fuente: Optimización de la geometría de una prótesis de miembro superior

Diseño de prótesis en el siglo XX

Para el siglo XX, el objetivo de que los amputados regresaran a su vida laboral, es alcanzado gracias a los esfuerzos del médico francés Gripoulleau, quien realizó diferentes accesorios que podían ser usados como unidad terminal, tales como anillos, ganchos y diversos instrumentos metálicos, que brindaban la capacidad de realizar trabajo de fuerza o de precisión.

En el año de 1912 Dorrance en Estados Unidos desarrolló el Hook, que es una unidad terminal que permite abrir activamente, mediante movimientos de la cintura escapular, además se cierra pasivamente por la acción de un tirante de

goma. Casi al mismo tiempo fue desarrollado en Alemania el gancho Fischercuya ventaja principal era que poseía una mayor potencia y diversidad en los tipos de prensión y sujeción de los objetos.

Es hasta 1946 cuando se crean sistemas de propulsión asistida, dando origen a las prótesis neumáticas y eléctricas. Un sistema de propulsión asistida es aquel en el que el movimiento es activado por algún agente externo al cuerpo.

Las prótesis con mando mioeléctrico comienzan a surgir en el año de 1960 en Rusia. Esta opción protésica funciona con pequeños potenciales extraídos durante la contracción de las masas musculares del muñón, siendo estos conducidos y amplificados para obtener el movimiento de la misma. En sus inicios, este tipo de prótesis solo era colocada para amputados de antebrazo, logrando una fuerza prensora de dos kilos

2.3. SISTEMAS PROTÉSICOS[3]

Toda prótesis artificial activa necesita una fuente de energía de donde tomar su fuerza; un sistema de transmisión de esta fuerza; un sistema de mando o acción y un dispositivo prensor. En la elección de las prótesis a utilizar desempeña un papel trascendental el nivel de amputación o el tipo de displasia de que se trate [2].

2.3.1. PRÓTESIS ELÉCTRICAS

Las prótesis eléctricas se basan en el uso de motores eléctricos, que pueden ser controlados por medio de servo-controles, pulsantes o interruptores, su principal desventaja es su reparación, su alto costo y su exposición a ambientes hostiles, así como también su peso. En la Figura 2.4 se puede observar una prótesis eléctrica

[3] Sistemas protésicos http://www.revista.unam.mx/

de la compañía Otto Bock que tiene como principal ventaja el agarre de objetos rápidamente y con precisión de forma activa gracias a los sensores en los dedos.

Figura 2.4: Prótesis eléctrica

Fuente: Dialnet – Estudio del estado arte de la prótesis de la mano

2.3.2. PRÓTESIS MIOELÉCTRICOS[4]

Las prótesis mioeléctricas son prótesis eléctricas controladas por medio de un poder externo mioeléctrico, estas prótesis son hoy en día el tipo de miembro artificial con más alto grado de rehabilitación. Sintetizan el mejor aspecto estético, tienen gran fuerza y velocidad de prensión, así como muchas posibilidades de combinación y ampliación (Figura 2.5).

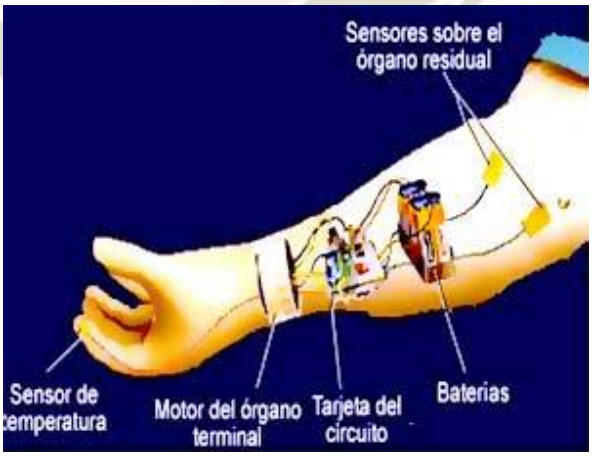


Figura 2.5: Configuración básica de una prótesis mioeléctricas

Fuente: Artículo de la revista de la UNAM

[4] Sistemas protésicos http://www.revista.unam.mx/

El control mioeléctrico es probablemente el esquema de control más popular. Se basa en el concepto de que siempre que un músculo en el cuerpo se contrae o se flexiona, se produce una pequeña señal eléctrica (EMG) que es creada por la interacción química en el cuerpo. Esta señal es muy pequeña (5 a 20 µV) Un micro-voltio es una millonésima parte de un voltio. Para poner esto en perspectiva, una bombilla eléctrica típica usa 110 a 120 voltios, de forma que esta señal es un millón de veces más pequeña que la electricidad requerida para alimentar una bombilla eléctrica.

El uso de sensores llamados electrodos que entran en contacto con la superficie de la piel permite registrar la señal EMG. Una vez registrada, esta señal se amplifica y es procesada después por un controlador que conmuta los motores encendiéndolos y apagándolos en la mano, la muñeca o el codo para producir movimiento y funcionalidad.

Este tipo de prótesis tiene la ventaja de que sólo requieren que el usuario flexione sus músculos para operarla, a diferencia de las prótesis accionadas por el cuerpo que requieren el movimiento general del cuerpo. Una prótesis controlada en forma mioeléctrica también elimina el arnés de suspensión usando una de las dos siguientes técnicas de suspensión: bloqueo de tejidos blandos-esqueleto o succión1. Tienen como desventaja que usan un sistema de batería que requiere mantenimiento para su recarga, descarga, desecharla y reemplazarla eventualmente. Debido al peso del sistema de batería y de los motores eléctricos, las prótesis accionadas por electricidad tienden a ser más pesadas que otras opciones protésicas. Una prótesis accionada por electricidad proporciona un mayor nivel de tecnología, pero a un mayor costo.

2.4. INTRODUCCIÓN A LA MIOGRAFÍA

Este capítulo describe brevemente la anatomía de los músculos, incluyendo el concepto de unidades motoras, seguido de la explicación de los diferentes métodos de medición común y aplicaciones propuso entender los movimientos físicos y fisiológicos de las señales generadas durante los movimientos musculares y son discutidos.

El cuerpo humano incluye varios sistemas como el sistema nervioso, el sistema cardiovascular y el sistema músculo-esquelético, que se compone de varios subsistemas que llevan a cabo muchos procesos fisiológicos complejos, por ejemplo, 'Acción', que puede ser mecánica, eléctrica o bioquímica. Actividades mecánicas se acompañan de señales que reflejan su naturaleza de las actividades en la forma de presión o la temperatura. Todo movimiento del cuerpo humano se realiza por el sistema muscular. Cualquier músculo se construye de órganos discretas que contienen tejidos esqueléticos musculares, vasos sanguíneos, tendones y nervios. Los tejidos musculares se subdividen en visceral, cardíaco y los tejidos óseos que realizan diferentes acciones conscientes e involuntarios.

En esta tesis nos centramos principalmente en los músculos esqueléticos por lo que los dos primeros grupos de tejidos no se discuten aquí. La mayoría de los músculos esqueléticos están unidos a dos huesos a través de tendones duros que se unen firmemente los músculos a los huesos. En el caso de los dedos, que no tienen los músculos y se mueven a través de los tendones unidos al antebrazo. Uno puede ver los movimientos de la superficie en su antebrazo cuando se mueven los dedos.

En esta tesis se estudia sobre los músculos del antebrazo que son responsables de los movimientos de los dedos y la palma. La razón para elegir el grupo muscular

del antebrazo es porque es de fácil acceso, las contracciones pueden ser fácilmente controladas y ajustadas y ha sido objeto de numerosos estudios en el pasado. Un diseño anatómico de los músculos de los brazos delanteros se puede ver en la figura 2.6. que muestra los tipos de músculos del antebrazo responsables de acciones diferentes compuestos.

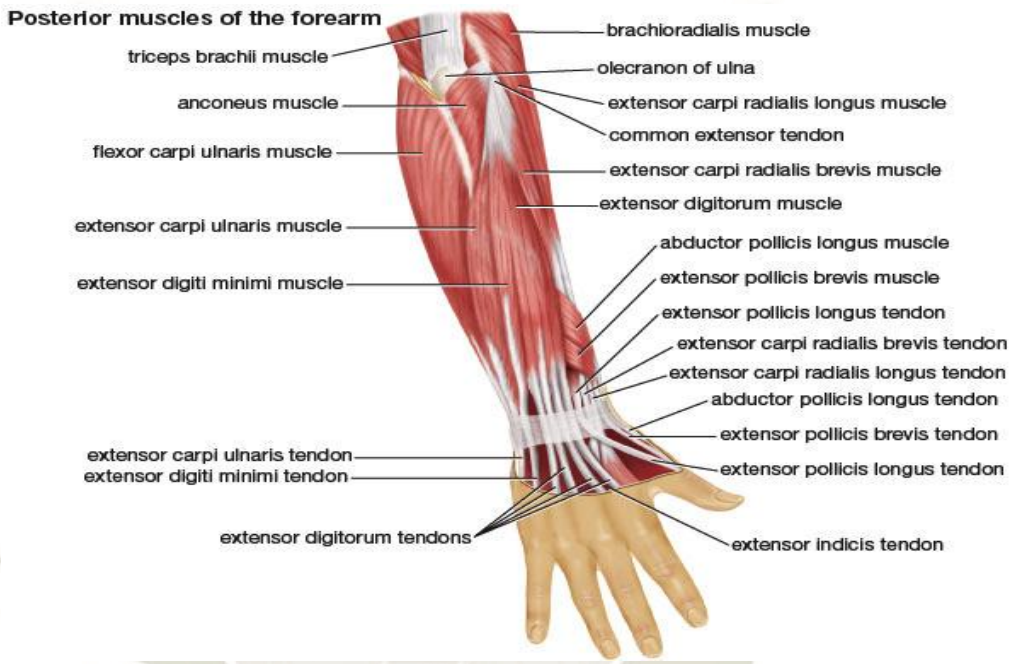


Figura 2.6: Grupos de músculos del brazo

Fuente: https://anatomy-library.com, partes de los músculos

Como se mencionó anteriormente, los dedos no tienen músculos en ellos. Ellos se mueven por los tendones, tirado por los músculos del antebrazo. Los músculos se pueden sentir cuando uno se mueve el dedo. Para 13 de cada movimiento diferente, ya sea por un dedo o por la totalidad diferente nivel de la muñeca de la fuerza se ejerce ya sea por uno o más grupos de músculos. Para girar todo el brazo sin mover la mano y la muñeca o para mover la muñeca hacia arriba y hacia abajo, la cantidad de fuerza es aplicada por los músculos posteriores, es decir, Extensor cubital del carpo, mientras poca fuerza se aplica en el músculo anterior, es decir, Flexor Carpi cubital. Como el antebrazo contiene muchos músculos en

la parte anterior y lateral posterior, nuestro principal objetivo fue observar los cambios en la superficie causados durante el movimiento de los dedos, la muñeca y el antebrazo y detectar esos cambios utilizando sensores fotoeléctricos.

2.5. MIOGRAFÍA

La medición de los fenómenos musculares tales como la velocidad y la intensidad de musculoso contracciones se llama miografía. Las oscilaciones laterales en los músculos, crean respectivas oscilaciones mecánicas en la superficie de la piel, son conocidos de largo. Diferentes técnicas se nombran de acuerdo a las mediciones que pueden ser cualesquiera actividades eléctricas o mecánicas producidas por los músculos esqueléticos. EMG, ECG y MMG son algunas de las técnicas más populares que se utilizan ampliamente para estudiar y comprender los defectos en el sistema humano utilizando electrodos o micrófonos o piezo-eléctricos sensores para registrar la actividad muscular.

2.5.1. LA ELECTROMIOGRAFÍA (EMG)

La demanda de métodos más avanzados de EMG superficie tiene una muy importante aplicación no sólo en el diagnóstico clínico, sino también para la ingeniería biomédica aplicaciones. Mediante el examen de la señal se puede entender la naturaleza y características de la señal que además ayuda en la rehabilitación de los trastornos o en la implementación de un hardware adecuado para diversas aplicaciones. Las fuerzas y las presiones aplicadas de unidad de motor potenciales de acción (MUAPs) en señales EMG proporcionan información detallada en el diagnóstico de los trastornos neuromusculares. El viejo e invasivo método de EMG utiliza electrodos que se insertan directamente en el músculo objetivo. La técnica no invasiva utiliza electrodos de superficie colocados sobre

la piel sobre el grupo muscular diana para registrar los potenciales de acción de fibra que se producen debajo de la piel.

2.6. OPTOMIOGRAFÍA

Es una nueva técnica propuesta en el año del 2015 que puede monitorear la actividad muscular.

Se trata de medir el desplazamiento muscular causado en la superficie de la piel debida su actividad (Figura 2.7.), con la introducción y desarrollo de sensores infrarrojos foto-eléctricos de proximidad a la cual se llamó optomiografía (OMG).

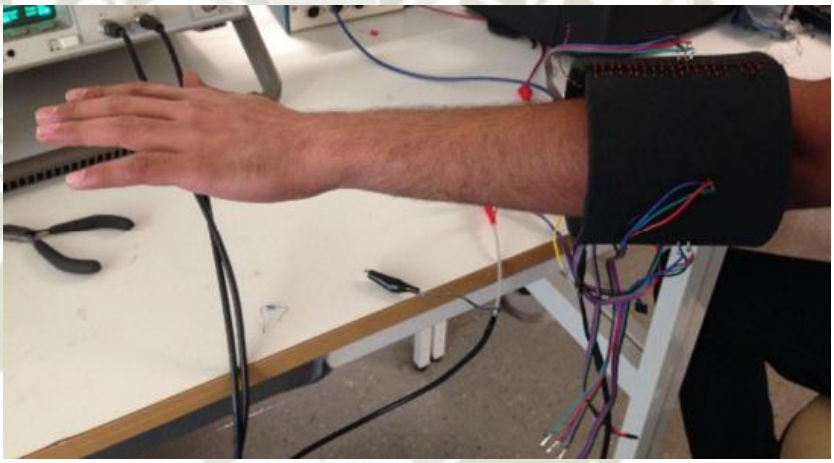


Figura 2.7: Banda del brazo de prueba de OMG.

Fuente: Optomyography (OMG): A Novel Technique for the Detection of Muscle Surface Displacement Using Photoelectric Sensors
Autor: Hamed Hamid Muhammed, Jammalamadaka Raghavendra

Con este dispositivo electrónico busca registrar las señales en tiempo real con el uso de sensores IR ubicados en el antebrazo mientras se mueve la mano haciendo diferentes posiciones, donde inicialmente ubicando dos sensores en posiciones opuestas. Con las dos diferentes señales que cambian en el tiempo nos da como resultado patrones repetitivos (Figura 2.8) mientras la misma posición de la

mano. Por lo cual se medirá diferentes posiciones y almacenaran los patrones de la mano[5] según el uso o estudio a emplear.

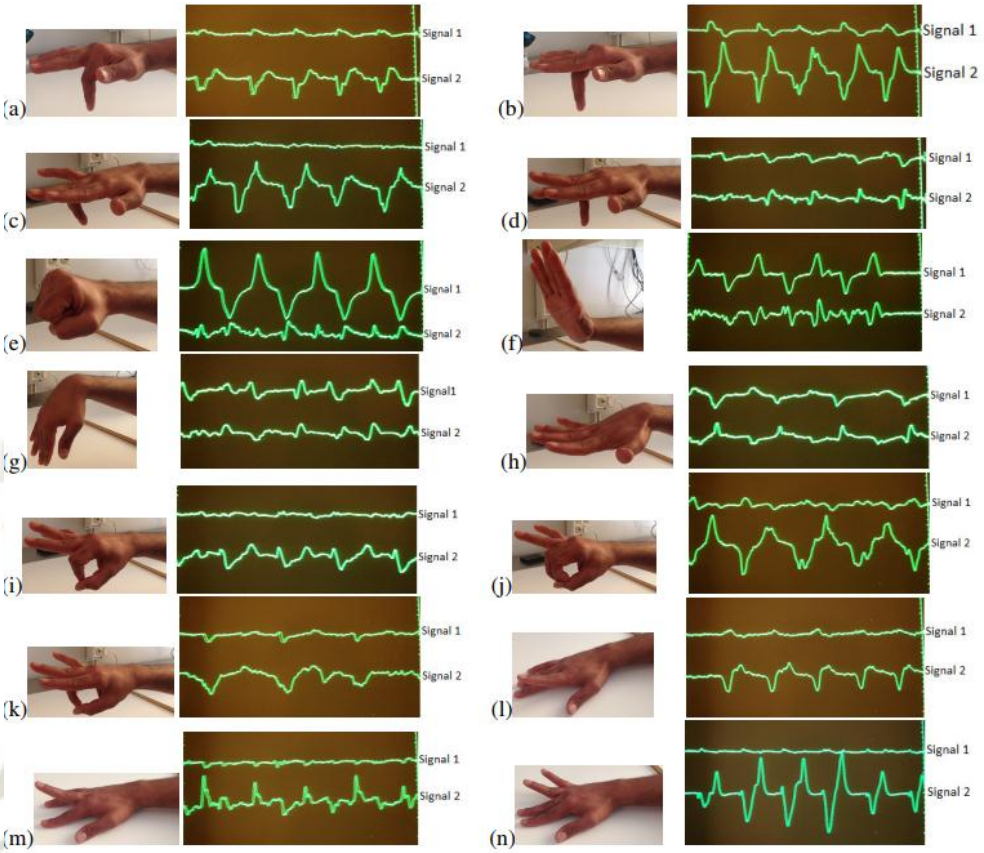


Figura 2.8: Catorce Patrones de desplazamiento.

Fuente: Optomyography (OMG): A Novel Technique for the Detection of Muscle Surface Displacement Using Photoelectric Sensors

2.7. SENSORES FOTOELÉCTRICOS

Esta sección se centra en la introducción de sensores fotoeléctricos, tipos y sus aplicaciones en el campo industrial y médico. Seguido de una breve descripción de los materiales utilizados generalmente para la construcción, junto con el procedimiento utilizado para el diseño. Aunque la mayoría de los sensores fotoeléctricos se utilizan en las aplicaciones industriales, hemos utilizado para detectar los cambios de desplazamiento sobre la superficie de la piel.

[5] Optomyography (OMG): A Novel Technique for the Detection of Muscle Surface Displacement Using Photoelectric Sensors

Los sensores fotoeléctricos en general son un dispositivo utilizado para detectar las variaciones en la intensidad de la luz, ya sea mediante la detección o no detección de la fuente de los sensores de luz. Estos sensores se componen de un LED, un fototransistor que actúa como un receptor, un convertidor de señal y un amplificador como se muestra en la figura 2.9.

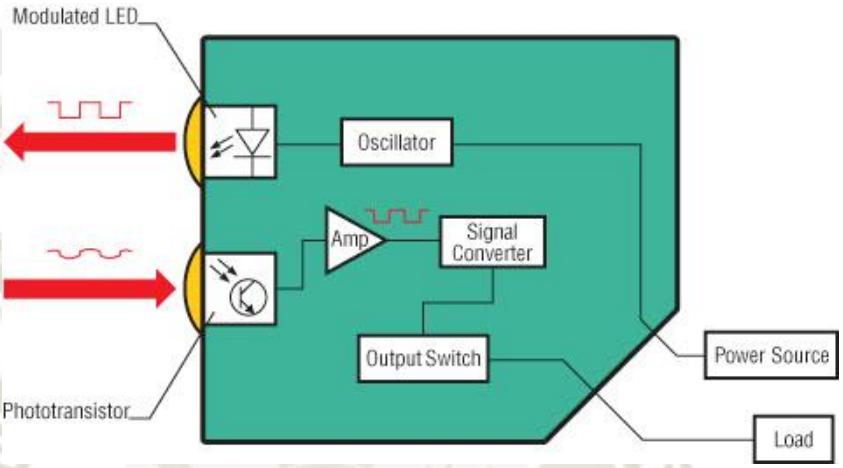


Figura 2.9: Principio de funcionamiento de sensores fotoeléctricos

Fuente: http://dominion.com.mx, sensores fotoeléctricos

Un haz de luz (visible o infrarroja) se emite desde el LED de transmisión que cuando se refleja desde la diana se detecta por el receptor [18].

2.7.1. SENSORES FOTOELÉCTRICOS

Basado en el tipo de modos de detección, sensores fotoeléctricos se clasifican en sensores de barrera, sensores retro-reflectantes, y sensores refractantes difusas.

2.7.1.1. Sensores retro-reflectantes

En los sensores de retro-reflectante, a diferencia de los sensores de barrera, el sensor consiste en el transmisor y el receptor incorporado en el mismo dispositivo, pero un reflector se utiliza para reflejar la luz de nuevo del transmisor al receptor. La detección se produce cuando un objeto bloquea la luz del transmisor al reflector como se muestra en la figura 2.10.

Figura 2.10: Sensor retro-reflectante

Fuente: http://dominion.com.mx, tipos de sensores fotoeléctricos

Aunque estos sensores tienen un eje óptico fácilmente ajustable y son fáciles de instalar, que restringen la instalación en un espacio limitado. Debido al uso de reflector de la eficacia de estos sensores son más eficaces que el sensor de modo difuso, lo que les permite censar un amplio rango.

Sensores retro-reflectantes con filtros polarizados que tienen un valor bajo de histéresis que les permite detectar pequeños cambios en la luz haciéndolos capaces de detectar objetos claros. Estos sensores cuando se opera dentro de una cierta distancia o zona muerta, eliminan el riesgo de falsa identificación de objetivos brillantes como reflector que es un error muy común durante el uso de sensor de modo retro-reflectante estándar.

2.7.2. FUNDAMENTOS LA ÓPTICA DE LA PERCEPTIVA

La precepción de la relación entre la luz del LED con varios materiales transparentes y semitransparentes, objetos de colores, objetos reflectantes y espejos que nos ayuda a comprender la reacción de los sistemas fotoeléctricos con los objetivos y el ruido de las condiciones operativas. Aunque la mayoría de los modelos fotoeléctricos tienen el mismo circuito eléctrico, teniendo de base, la elección de los elementos de fotos y su ubicación con respecto a la lente, que están diferenciadas como la proximidad, retro-reflectante o modo difuso. Estos cambios crean diferentes formas de haz y las direcciones que son responsables de diferentes características de detección.

2.8. MÁQUINA DE ESTADO FINITO

También conocida con las siglas en ingles de FSM (*Finite State Machine*) y FSA (*Finite State Automation*), son modelos del os comportamientos de un sistema o un objeto complejo, con un número limitado de condiciones o modos definidos, donde las transiciones de modo que cambian según las circunstancias. [6]

La máquina de estado finito consta de cuatro elementos principales:

- **Estados:** Este define el comportamiento y puede producir acciones.
- **Transición de estados:** El cual se mueve de un estado a otro estado.
- **Reglas o condiciones:** Se debe cumplir para la transición de un estado a otro.
- **Eventos de entrada:** los cuales pueden ser generadas externamente e internamente, los cuales tienen la posibilidad de manejar las reglas a los estados de transiciones.

Una máquina de estados finitos debe tener un estado inicial que proporcione un punto de partida y un estado actual que recuerde el producto de la última transición de estado. Los eventos de entrada recibidos actúan como activadores, lo que provoca una evaluación de algún tipo de reglas que rigen las transiciones del estado actual a otros estados. La mejor manera de visualizar un FSM es considerarlo como un diagrama de flujo o un gráfico de estados dirigido, aunque como se mostrará[13]. Existen técnicas de modelado abstracto más precisas que se pueden utilizar (figura 2.11).

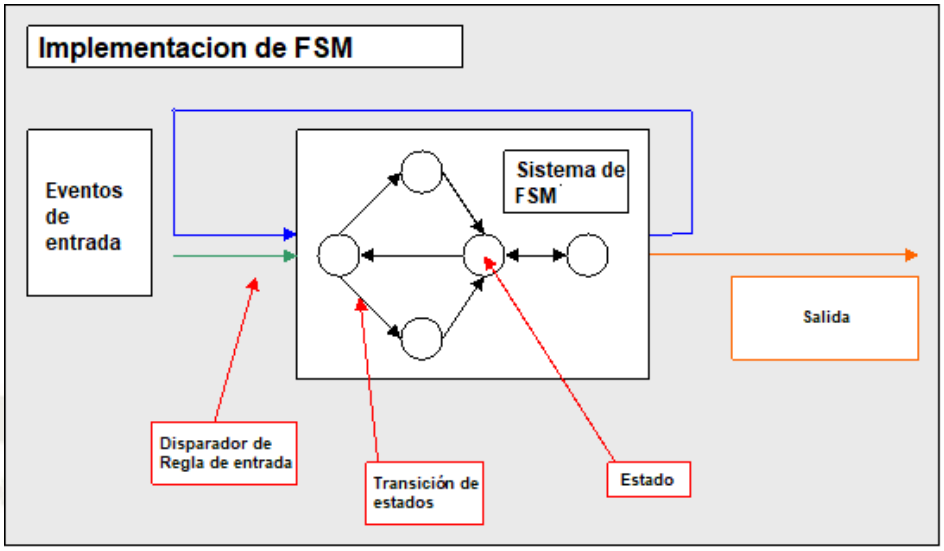


Figura 2.11: Máquina de estados finitos

Fuente: http://ai-depot.com, máquina de estados finitos

2.8.1. Ventajas del FSM

Su simplicidad facilita que los desarrolladores sin experiencia implementen con poco o ningún conocimiento adicional (bajo nivel de entrada)

La previsibilidad (en FSM determinista), dado un conjunto de entradas y un estado actual conocido, la transición de estado puede predecirse, lo que permite una prueba fácil

Debido a su simplicidad, las FSM son rápidas de diseñar, rápidas de implementar y rápidas de ejecución

FSM es una técnica antigua de representación de sistemas y modelos de sistemas, y ha existido durante mucho tiempo, por lo que está bien probada incluso como una técnica de inteligencia artificial, con muchos ejemplos de los cuales aprender.

Los FSM son relativamente flexibles. Existen varias formas de implementar un sistema basado en FSM en términos de topología, y es fácil incorporar muchas otras técnicas.

Fácil de transferir desde una representación abstracta significativa a una implementación codificada

Baja sobrecarga del procesador; adecuado para dominios donde el tiempo de ejecución se comparte entre módulos o subsistemas. Solo se necesita ejecutar el código para el estado actual, y quizás una pequeña cantidad de lógica para determinar el estado actual.

La fácil determinación de la accesibilidad de un estado, cuando se representa en una forma abstracta, es inmediatamente obvia si un estado es alcanzable desde otro estado, y qué se requiere para alcanzar el estado

2.8.2. Desventajas de FSM

La naturaleza predecible de los FSM deterministas puede no ser deseada en algunos dominios, como los juegos de computadora (la solución puede ser un FSM no determinista).

Los sistemas más grandes implementados utilizando un FSM pueden ser difíciles de administrar y mantener sin un diseño bien pensado. Las transiciones de estado pueden causar un grado considerable de "factor de espaguetis" cuando se intenta seguir la línea de ejecución

No es adecuado para todos los dominios de problemas, solo debe usarse cuando un comportamiento de los sistemas se puede descomponer en estados separados con condiciones bien definidas para las transiciones de estado. Esto significa que todos los estados, transiciones y condiciones deben ser conocidos por adelantado y bien definidos.

Las condiciones para las transiciones de estado se superan, lo que significa que son fijas (esto se puede superar utilizando una máquina de estado difusa (FuSM))

Como la mayoría de las técnicas, las heurísticas de cuándo y cómo implementar máquinas de estados finitos son subjetivas y específicas de los problemas. Está claro que los FSM son adecuados para los dominios de problemas que se expresan fácilmente mediante un diagrama de flujo y poseen un conjunto de estados y reglas bien definidos para controlar las transiciones de estado.

Para una discusión más amplia de las máquinas de estados finitos, te recomiendo que leas; Máquinas de estado finito: hacer trabajos simples de funciones complejas.

2.9. MICROCONTROLADOR TM4C123GH6PM

El TM4C123GH6PM de Texas Instruments es un microcontrolador de alto rendimiento de 32-bits con una arquitectura ARM Córtex-M y una amplia cantidad de herramientas para el desarrollo de aplicaciones.

El hecho de que este microcontrolador se encuentre diseñado sobre una arquitectura ARM Córtex-M implica que el procesador proveerá alto rendimiento, bajo costo, mínimo uso de memoria, uso reducido de pines y bajo consumo de potencia, mientras brinda un alto rendimiento computacional y una respuesta excepcional ante interrupciones.

2.9.1. Características

Son los datos relevantes que presentan el microcontrolador del módulo, con lo cual entendemos que el sistema se encuentra sobre dimensionado para el uso dado.

Tabla N° 1: Especificaciones del TM4C123GH6PM

CARACTERÍSTICAS	DETALLES
RENDIMIENTO	
Núcleo	ARM Cortex-M4F
Rendimiento	80-Mhz; 100 DMIPS
FLASH	265 KB de ciclo único
SRAM	32 KB de ciclo único
EEPROM	2KB
ROM interna	Programado con TIVAWARE
INTERFACES DE COMUNICACIÓN	
Universal asynchronous Receivers/transmitter (UART)	8 módulos
Synchronous serial interface (SSI)	4 módulos
Inter-integrated circuit (I2C)	4 módulos con 4 velocidades de transmisión incluido modo de alta velocidad.
Controller area network (CAN)	2 controladores CAN 2.0 A/B
Universal serial bus (USB)	USB 2.0. OTG/Host/Device
INTEGRACIÓN DEL SISTEMA	
Micro direct memory Access (µDMA)	32 canales
General-purpose timer (GPTM)	Seis contadores de 16/32 bits y seis contadores de 32/64 bits
Watchdog timer (WDT)	2 módulos
Hibernation module (HIB)	Módulo de bajo consumo
General-purpose input/output (GPIO)	6 bloques físicos
CONTROL AVANZADO DE MOVIMIENTO	
Pulse width modulator (PWM)	2 módulos, cada uno con cuatro generadores, haciendo un total de 16 salidas PWM
Quadrature encoder interface (QEI)	2 módulos
SOPORTE ANALÓGICO	
Analog-to-digital converter (ADC)	2 módulos de 12bits de resolución con una tasa de conversión de 1 millón de muestras por segundo cada uno
Comparador analógico	2 compradores analógicos independientes integrados
Comparador digital	16 comparadores
JTAG *and serial wire debug* (SWD)	Un módulo JTAG con ARM serial wire debug integrado

Fuente: Hoja de datos del TM4C123GH6PM

2.9.2. ENERGÍA IDE

ENERGÍA.NU es una IDE que se basa en el popular programa de PROCESSING o ARDUINO que es fácil de usar. Esta famosa interfaz de programación en "C", ensamblador y código basado en manejo de librería.

Monitor serial integrado

Se comunica con el módulo de prueba vía serial. La energía ofrece la construcción de un monitor serial. El terminal de habilita una comunicación bidireccional para el módulo electrónico a diferencias ratios de baudios.

Figura 2.12: Interfaz Energia.NU

Fuente: Elaboración propia

2.10. PROGRAMACIÓN LABVIEW

El programa LABVIEW es un revolucionario sistema de programación grafica que sirve para aplicaciones que requieran adquisición, control, análisis y representación de data.

Este programa reduce el tiempo del desarrollo de aplicaciones dentro de un rango de 4 a 10 veces, ya que nos da la facilidad de aprendizaje debido a una peculiaridad que es ser muy intuitivo.

También debido a su gran adaptación de hardware y dispositivos, está dotado de gran flexibilidad que permiten cambios en lo antes mencionado.

Al programador le da la facilidad de poder crear diferentes soluciones completas y complejas según sea requerido.

Su desarrollo integral de funciones de adquisición, análisis y presentación de datos. También debido a su flexibilidad da la posibilidad de incorporar diversos lenguajes de programación.

LABVIEW es un programa que requiere básicamente la programación gráfica, también conocido como lenguaje de programación G, ya que su programación se basa en la creación de bloques.

Una de las grandes ventajas del programa de LABVIEW es que no se requiere gran experiencia en programación, ya que emplea terminologías que son familiares para nuestro rubro; es decir; par ingenieros y científicos. Ya que sus cimientos son la simbología grafica en ves del escrito para la creación de aplicaciones. Debido a esto es más intuitivo que los demás lenguajes de programación.

Este programa presenta una gran variedad de librerías de funciones y subrutinas, pre-desarrolladas para facilitar la programación y depuración de los programas a desarrollar o desarrollado, dependiendo de la fase en la cual se encuentre.

2.10.1. Instrumentos virtuales

Los programas que desarrollan en el entorno LABVIEW son conocidos como instrumentos virtuales (VI's), debido a su apariencia y a que imitan los de uno real.

Los VIs tienen dos partes, una que interactúa con el usuario y otra que parte que es el código fuente.

Dicho lo anterior este tiene un panel frontal y un diagrama de bloques.

a. Panel Frontal

Esta interfaz recoge las entradas procedentes del usuario y representa las salidas proporcionadas por el programa. Un panel frontal está formado por una serie de botones, pulsadores, potenciómetros, gráficos, etc.

Cada uno de ellos puede estar definido como un control (a) o un indicador (b). Los primeros sirven para introducir parámetros al VI, mientras que los indicadores se emplean para mostrar los resultados producidos, ya sean datos adquiridos o resultados de alguna operación (Figura 2.13).

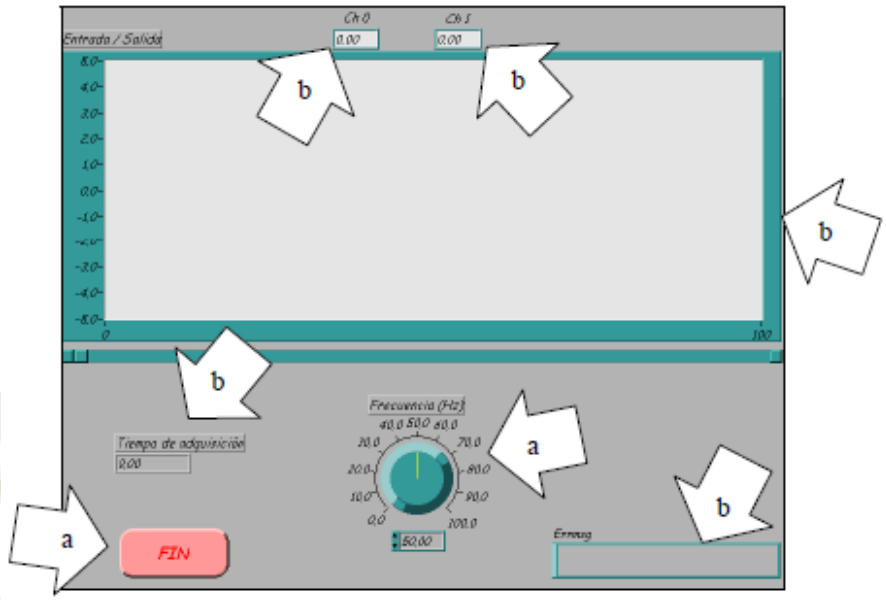


Figura 2.13: Panel frontal VI

Fuente: www.esi2.us.es, Tutorial de Labview

b. Diagrama de bloques

El diagrama de bloques constituye el código fuente del VI. En el diagrama de bloques es donde se realiza la implementación del programa del VI para controlar o realizar cualquier procesado de las entradas y salidas que se crearon en el panel frontal.

El diagrama de bloques incluye funciones y estructuras integradas en las librerías que incorpora LABVIEW. En el lenguaje G las funciones y las estructuras son nodos elementales. Son análogas a los operadores o librerías de funciones de los lenguajes convencionales. (Figura 2.14)

Los controles e indicadores que se colocaron previamente en el Panel Frontal, se materializan en el diagrama de bloques mediante los terminales. A continuación, se presenta un ejemplo de lo recién citado:

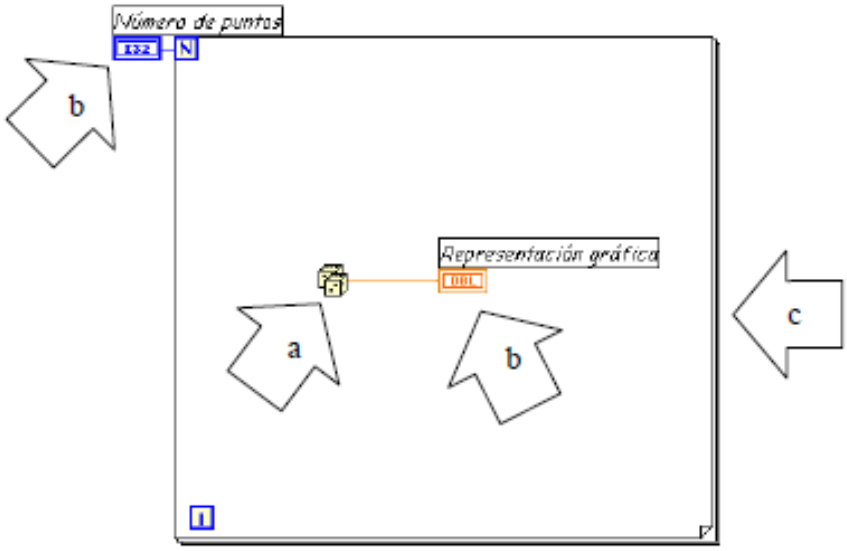


Figura 2.14: Panel de diagrama de bloques

Fuente: www.esi2.us.es, Tutorial de Labview

(a) Función.

(b) Terminales (control e indicador).

(c) Estructura.

El diagrama de bloques se construye conectando los distintos objetos entre sí, como si de un circuito se tratara. Los cables unen terminales de entrada y salida con los objetos correspondientes, y por ellos fluyen los datos.

LabVIEW posee una extensa biblioteca de funciones, entre ellas, aritméticas, comparaciones, conversiones, funciones de entrada/salida, de análisis, etc. Las estructuras, similares a las declaraciones causales y a los bucles en lenguajes convencionales, ejecutan el código que contienen de forma condicional o repetitiva (bucle for, while, case...).

CAPITULO III
DESARROLLO DE LA INGENIERÍA

CAPITULO III

DESARROLLO DE LA INGENIERÍA

3.1. DIAGRAMA DE BLOQUES

Primero desglosaremos de forma básica, las señales de entradas y salidas en una caja negra donde la primera señal de entrada es la contracción muscular y se desea obtener en la salida el desplazamiento de los dedos en la mano robótica y la segunda señal está dada por la alimentación que indirectamente nos generará un leve ruido y vibración por el funcionamiento de los servomotores y perdidas parásitas, más común en las resistencias durante el consumo (Figura 3.1.).

Figura 3.1: Caja negra del proyecto

Fuente: Elaboración propia

El sistema se basa en el uso de un microcontrolador, donde se desarrollará la lógica de control además de la adquisición y procesamiento de las señales. Primero un sensor infrarojo recibe la información que es recolectada debido al desplazamiento muscular, luego envía los datos recolectados al microcontrolador para ser procesada y finalmente entregar la señal de control al driver de los servomotores para luego realizar el desplazamiento en conjunto con mecanismo de barras y obtener la trayectoria de desplazamiento del efector final.

3.1.1. ETAPAS DEL SISTEMA: PRINCIPALES

El control por OMG esta ordenado en etapas. El camino contiene tres etapas principales y una de monitoreo:

1. **Etapa 1:** Contracción Muscular
2. **Etapa 2:** Procesamiento de Señal
3. **Etapa 3:** Movimiento de los dedos
4. **Etapa 4:** Interfaz Virtual (Monitoreo)

Esto será mostrado de forma práctica y básica en el siguiente diagrama de bloques (Figura 3.2).

Donde el usuario en la etapa uno realizará la contracción muscular con su musculo pronador y este desplazamiento será captado por el sensor IR convirtiendo esta señal en voltaje que será procesada en el microcontrolador, finalmente a la mano robótica le llegará la señal de control a los servomotores y realizará el desplazamiento de los dedos.

La explicación más detallada se encontrará en la sección 3.2.

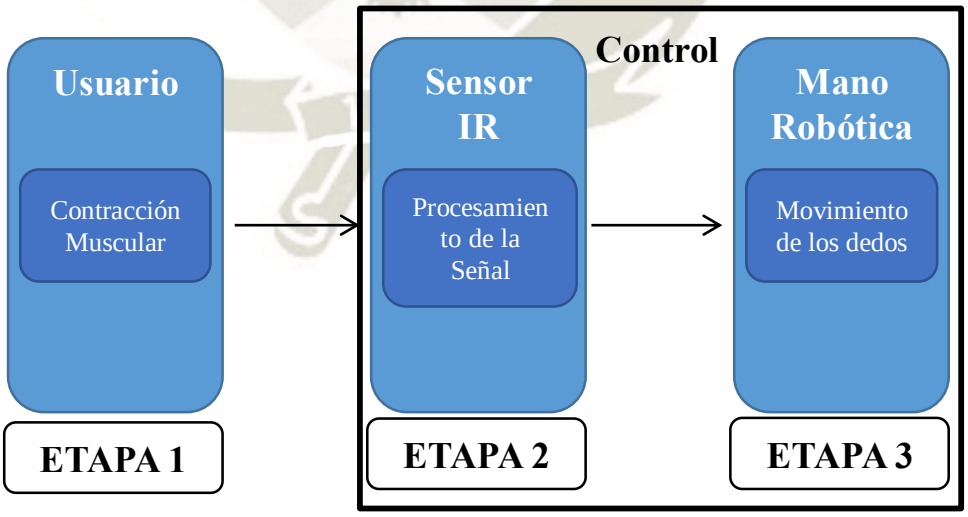


Figura 3.2: Las 3 etapas básicas para el control

Fuente: Elaboración propia

3.1.2. PROGRAMACIÓN: DISEÑO DEL SISTEMA DE CONTROL

En el desarrollo de la programación, de tiene como requerimiento básico definir una estructura y una lógica de control, la lógica que se diseñó para el sistema de control será explicado a continuación, tomando como base la figura 3.3. que es el resumen del sistema de control que se verá y explicará en la sección 3.3. donde se analizará las variaciones reales de la señal.

Primero es necesario realizar la calibración y que se ejecute una vez en el programa para establecer los valores máximos y mínimos de voltaje que se puede recibir al realizar la contracción muscular, después haber instalado la banda con el sensor IR, al haber obtenido el rango de operación, es necesario establecerlos como límites de saturación y corte.

Al finalizar la calibración es necesario hacer un escalado para un manejo más práctico de las variables por lo cual se realiza una conversión porcentual a través de un mapeo o la aplicación del teorema de tales de los límites de saturación y corte.

Para facilitar el control se creó un nivel de referencia, con el cual lograremos convertir la señal porcentual a valores booleanos es decir que trabajaremos con 1’s y 0’s al terminar esta etapa del procesamiento.

Luego pasamos con estos valores booleanos a la detección del tipo de señal, con la cual identificaremos los tipos de pulsos generado por usuario establecidos en un periodo determinado, al finalizar esta etapa podremos detectar y generar como salida las señales de Co-contracción(CC), Doble

impulso(DI), Triple impulso(TI), Mantener-soltar(HO) y Doble mantener-soltar(DHO).

Ya con estas señales podemos controlar la lógica de control que fue desarrollada en diagramas de estados y desarrollado por mi autoría para manejar ocho posibles combinaciones de desplazamiento. Con el diagrama de estados principal se podrá realizar transición en los cuatro estados (Teniendo como señales de control a CC, DI, TI) y cada estado posee teóricamente un sub-diagrama de estados con dos lógicas y así poder realizar las combinaciones preestablecidas.

En la salida se desarrollará un vector con condiciones de desplazamiento para cada servomotor (S1, S2, S3) que llegará a una función y convertirá estos a valores angulares para cada servomotor.

Para obtener como resultado que alguna función o lógica en el programa interrumpa a la otra función por lo cual se requiere la realización de multitarea o programación "*multitasking*" debido que la mayoría de las variables son dependientes del tiempo.

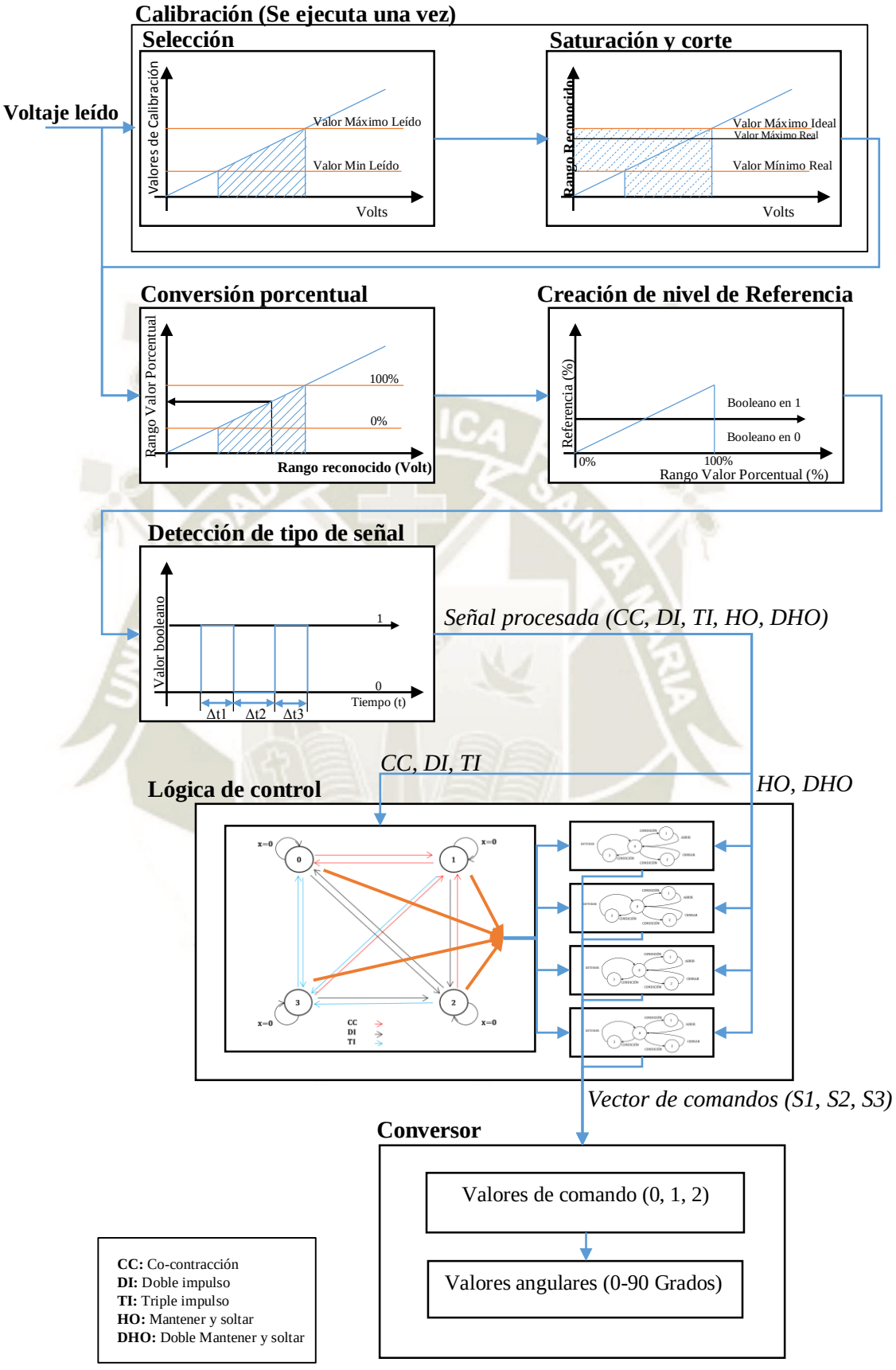


Figura 3.3: Diagrama resumen del diseño del sistema de control

Fuente: Elaboración propia

3.1.3. HARDWARE: ESTRUCTURA DE FUNCIONAMIENTO FÍSICO

La estructura de funcionamiento posee un desglose del cómo se enviará las diversas señales y la interacción de las variables de entrada y salida durante el desarrollo de proyecto.

Al haber elaborado la caja negra se logra proseguir a realizar el desglose de la estructura de funcionamiento (Figura 3.1.).

En el microcontrolador se desarrollará la calibración, el procesamiento de la señal, creación, recepción y envío de los diversos tipos de señales, como por ejemplo la recepción de variación del voltaje que entrega el módulo del sensor IR, también la señal de comunicación con el con programa LABVIEW, que es a través del protocolo de comunicación RS232 y finalmente enviar la señal PWM de control a los servomotores.

Sabiendo que los servomotores presentan engranajes internos para realizar las transmisiones de potencia debido a este sistema de reducción nos entrega un desplazamiento angular que en la mano convertirá estos en desplazamiento angular a lineal en un servomotor y en los otros obtendremos cambio angular a angular debido a el mecanismo de barras y finalmente realizar el desplazamiento de los dedos.

Finalmente, durante la etapa de potencia, se realizará el acondicionamiento de señal de alimentación según lo que se desea alimentar, en nuestro sistema se identificó 3 requerimiento de acondicionamiento, primero la alimentación que requiere el sensor IR, luego la alimentación de la tarjeta electrónica con el microcontrolador y por último la alimentación para cada servomotor (Figura 3.4.).

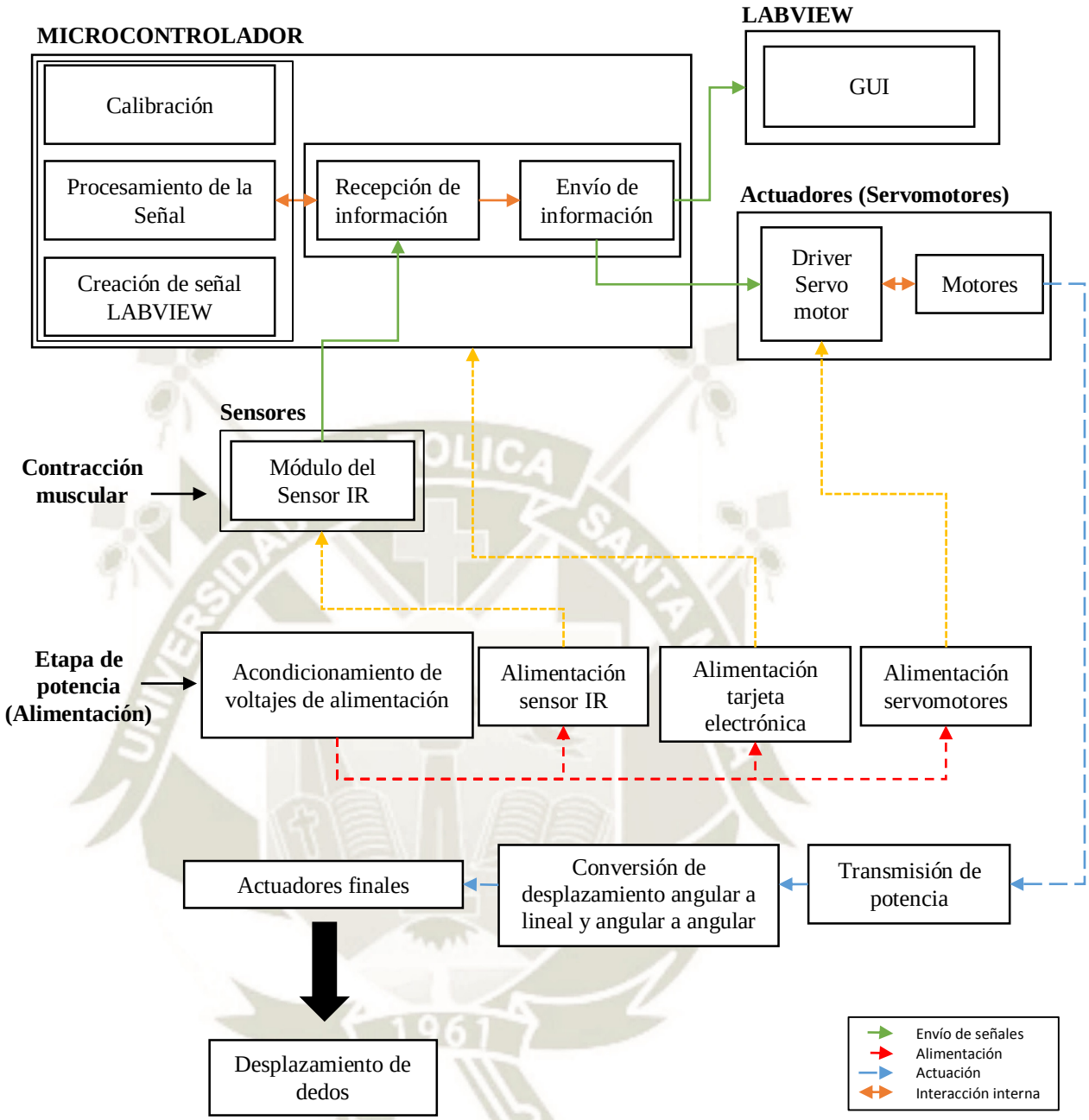


Figura 3.4: Estructura de funcionamiento

Fuente: Elaboración propia

3.2. ETAPAS DEL SISTEMA

En esta sección se explicarán las etapas y sus secuencias de funcionamiento lógico del sistema, que va desde la interacción del usuario hasta la ejecución del control de los actuadores finales. Durante el proceso obtendremos las lógicas de funcionamiento, el diseño de la programación lógica y la selección e implementación del hardware físico.

Para poder explicar cómo se desarrolló el sistema de control, este se ordenó en cuatro etapas, dando un orden de implementación en forma secuencial del proyecto, en la etapa cuatro se desarrolló el sistema de monitoreo donde se podrán visualizar de todas las variables adquiridas como las de control para la verificación del correcto funcionamiento en caso de ser necesario.

En la **etapa uno**, se inicia por el usuario. El usuario realiza la contracción de los músculos para poder obtener la señal. Esta acción crea una señal de voltaje la cual es transferida de la etapa uno y luego a la etapa dos.

En la **etapa dos**, la señal que es adquirida por el sensor IR que el encargado de leer la variación de voltaje, en esta etapa se marca un nivel de referencia para poder realizar la detección del tipo de señal de control y luego convertida a valores digitales para que el microcontrolador implemente la lógica en la siguiente etapa.

En la **etapa tres,** en esta etapa se desarrollará lógica de control, donde se logrará que interactúe el proyecto de forma funcional, es decir, que procesara la información adquirida y enviará las señales de control a los actuadores para luego poder desplazar el efector final.

En la **etapa cuatro,** se desarrollará la programación de adquisición de datos y la interfaz gráfica usuario en el software LABVIEW.

Estas cuatro etapas serán detalladas en esta sección. Resumiremos los pasos en una la figura 3.5.

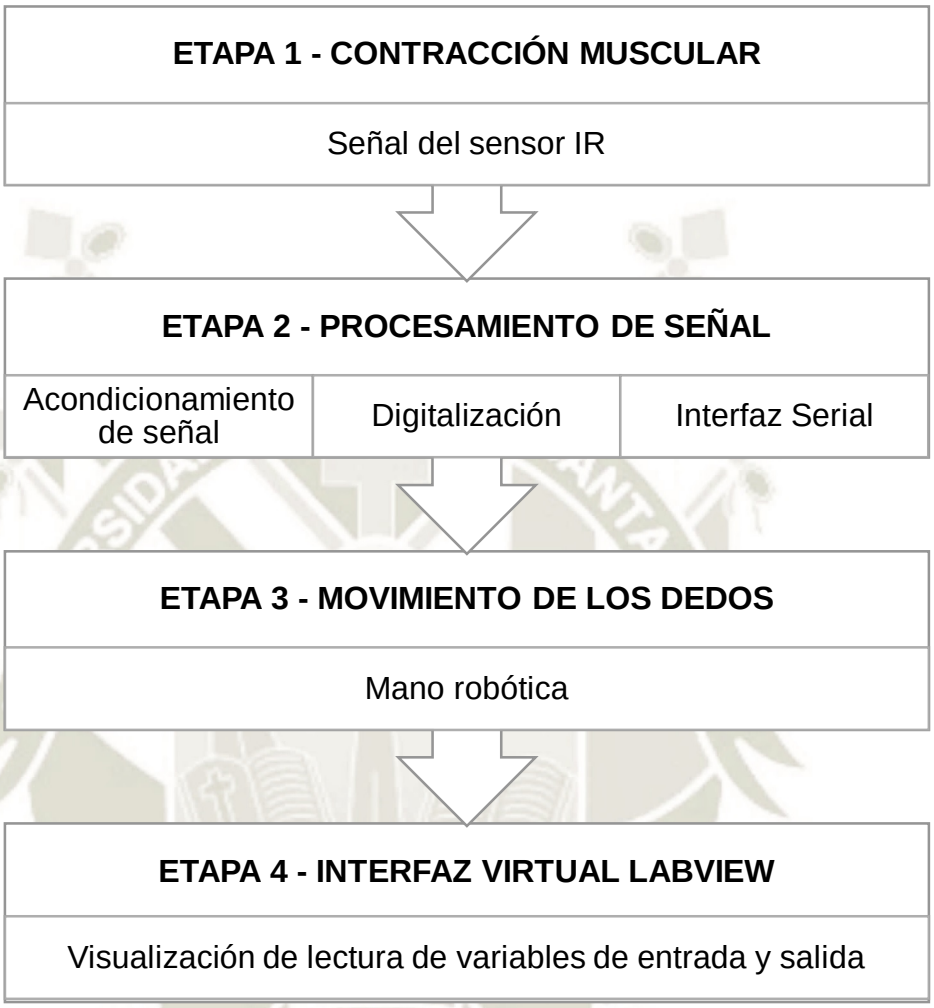


Figura 3.5: Flujo de las etapas del sistema

Fuente: Elaboración propia

3.2.1. ETAPA 1 – Contracción muscular

Un sensor óptico que emite luz con una longitud de onda que se dispersa por la piel que, dependiendo de la longitud de onda, las propiedades ópticas de capas de la piel determinan las propiedades de reflexión y absorción de la misma. La capa de epidermis de la piel se compone de melaninas y queratinas. Melaninas tienen altos índices de refracción, que dispersa la luz en un alto grado[6]. Las propiedades reflectantes de la piel aumentan con el aumento de la longitud de onda de la luz incidente, como se muestra en la Figura. 3.6. [17]

Efectos en la contracción muscular

Esta es una alternativa no invasiva alternativa al tradicional Electrodos de EMG que podemos monitorearla.

Fondo biológico

Con este sensor nos enfocamos en las contracciones isométricas, las contracciones en cuanto a la longitud del músculo esta no cambia cuando la fuerza es aplicada. Por ejemplo, apretando el puño provoca una contracción isométrica del bíceps

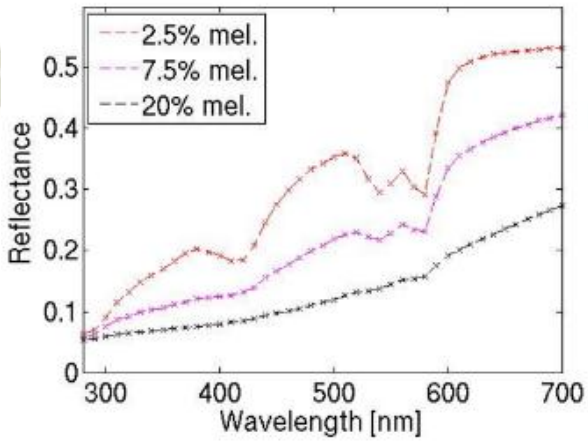


Figura 3.6: Propiedades reflectantes de la piel dependiendo del ancho de onda

Fuente: The optics of human skin: Aspects important for human health

[6] http://www.dnva.no, Nr 3 Nielsen

La adquisición se realiza en tres partes:

Un alargamiento rápido del bíceps, un periodo de fuerza constante muscular y longitud, y luego la liberación lenta del musculo. Durante el alargamiento inicial el musculo, las fibras de actina y miosina en el musculo se estiran. Esto reduce la reflectancia de la luz IR del músculo. Durante la liberación del musculo, la sangre fluye de forma relativamente lenta de nuevo al músculo, dando una mayor reflectancia de la luz IR del músculo, debido al flujo de sangre.

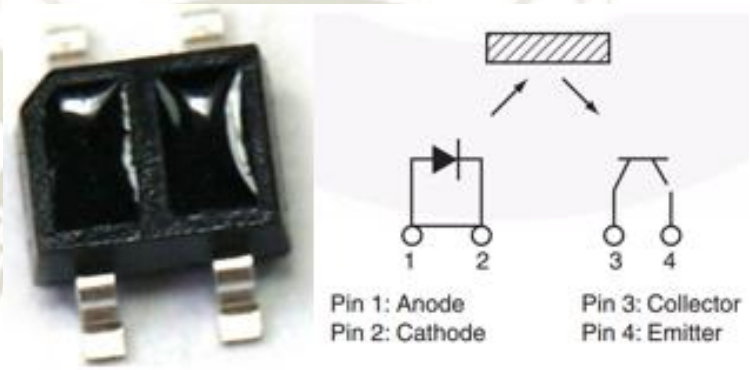


Figura 3.7: Sensor QRE1113

Fuente: Hoja de datos del sensor QRE1113

El foto-diodo del QRE1113 (Figura 3.7) emite luz infrarroja de 940nm de longitud de onda y una alta sensibilidad del detector, esto hace que este sea una los principales motivos además de su pequeño tamaño para su selección.

El transistor genera una señal de salida cuando se detecta la fuente de luz reflejada de un objetivo. También cuenta con un filtro de luz visible incorporado que reduce al mínimo la influencia de la luz externa.

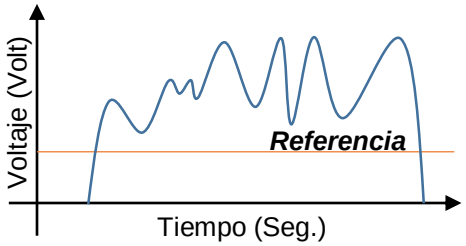


Figura 3.8: Voltaje leído

Fuente: Elaboración propia

En esencia, hay dos requerimientos para transmitir la señal correcta leída (figura 3.8) a la etapa 2. Primero, el sensor debe ubicarse en un lugar óptimo para obtener la mejor lectura del sensor.

En segundo, el usuario debe ejecutar de forma correcta las contracciones para producir las señales.

Cumpliendo esto la señal será transmitida de forma adecuada a la etapa 3.

3.2.2. ETAPA 2 – Procesamiento de Señal

Para obtener la señal deseada y procesada se requiere una secuencia de pasos, que se ordenaron de forma secuencial, para poder explicar de forma práctica como se desarrolló de paso a paso. (Figura 3.9.)

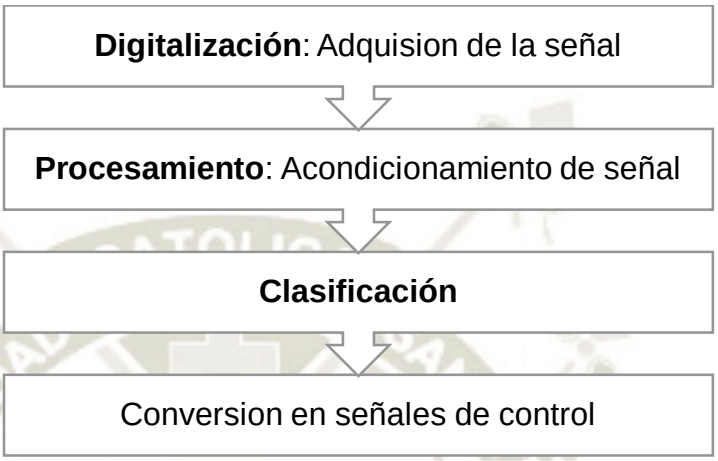


Figura 3.9: Etapas del sistema

Fuente: Elaboración propia

Hay 4 etapas de procesamiento para lograr el control de una mano robótica desde la señal adquirida por el EMG expresadas en la parte inferior del 1 al 4, pero no son necesarios para el control por sensores IR, reduciendo el nivel de complejidad para nuestro sistema de control y acondicionamiento del mismo.

1. Un filtro pasa-bajos para reducir el ruido.
2. Rectificación de la señal.
3. Un filtro pasa-bajos para crear una señal suave.
4. Amplificar la señal de rango a unos pocos voltios.

En los sistemas de control para prótesis en la actualidad existen 4 tipos de señales para controlarlos, los cuales usaremos como base para realizar nuestro control, se obtendrán en la etapa de clasificación conocidas como "*triggers"* o pulsos:

1. CC: Co-contracción
2. HO: Mantener-abrir.
3. DI: Doble impulso
4. TI: Triple impulso.

La explicación detallada se dará en la tabla 2. Se explicará cómo se aplica y el respectivo funcionamiento de cada señal.

Las entradas y salidas de esta etapa son las siguientes:

Tabla N° 2: Cuatro "TRIGGER" comunes de EMG para aplicación en IR control. Los desencadenantes descritos son co-contracción (CC), mantener-abrir (HO), doble impulso (DI) y triple impulso (TI).

TRIGGER	DESCRIPCIÓN DE LA SEÑAL OMG / EMG	ACCIÓN DEL USUARIO
CC	Las señales de flexión y extensión suben por encima del umbral dentro del período de tiempo de detección.	Contracción de ambos músculos simultáneamente.
HO	La mano está estancada abierta y la señal de extensión se mantiene por encima del umbral durante el tiempo de detección definido.	Contraiga el músculo extensor para abrir la mano. Relaje el músculo extensor.
DI	La mano está estancada abierta, la señal de extensión es 2 impulsos cortos con relajación por debajo de un umbral entre cada uno. Cada impulso debe cruzar el umbral.	Contraiga el músculo extensor dos veces rápidamente.
TI	La mano está estancada abierta, la señal de extensión es 3 impulsos cortos con relajación por debajo de un umbral entre cada uno. Cada impulso debe cruzar el umbral.	Contraiga el músculo extensor 3 veces rápidamente.

Fuente: VILARINO-THESIS-2013[7] [15]

Antes de la rectificación para cada TRIGGER para poder obtener las señales se requiere que cumpla unas condiciones que sean necesarias para que supere el

1. [7] Enhancing the Control of Upper Limb Myoelectric Prostheses using Radio Frequency Identication by Martin Vilarino

umbral de voltaje y un intervalo de tiempo necesario para que se considere valida la operación.

3.2.2.1. Digitalización:

Tras una etapa de acondicionamiento adecuado, las señales análogas del sensor son digitalizadas, transformadas y para su posterior análisis. Es importante mencionar, que la tasa de muestreo durante la adquisición de datos debe ser al menos dos veces la frecuencia más alta en la respuesta del sensor. Para que el proceso de conversión analógico-digital sea válido se deberá realizar al doble de la frecuencia máxima, a este proceso se le conoce como el teorema de muestreo de Nyquist, estas siendo preconfiguradas en el microcontrolador usado TM4C123GXL el cual tiene una frecuencia de muestreo de 2MSPS en los ADC's del mismo (Figura 3.10).

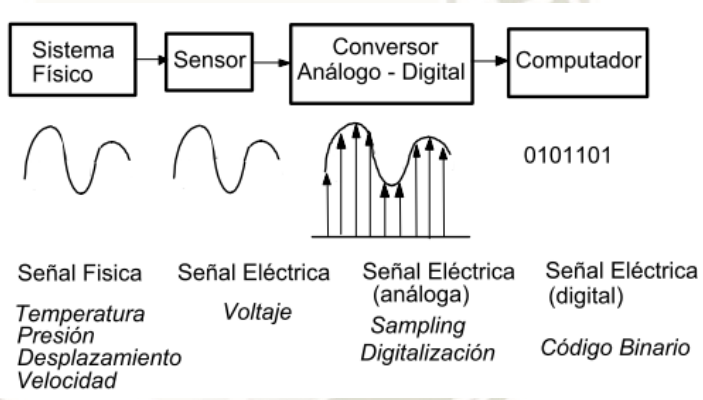


Figura 3.10: Comportamiento básico de un convertidor ADC

Fuente: https://22xd.blogspot.com

El microcontrolador maneja 12 bits de resolución en su ADC, es decir, que leerá los valores de voltaje de 0 a 3.3V en valores del microcontrolador lee en un rango de 0 a 4095. Nuestro sensor por experiencia entrega un valor máximo debido a su configuración que va del **0 a 1120** bytes de lectura que seria 0.9VDC de valor real de lectura. En la figura 3.11 se podrá visualizar los valores digitalizados en la variación del tiempo ya digitalizados.

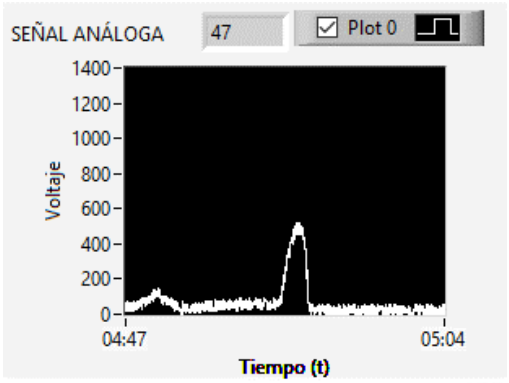


Figura 3.11: Visualización de parámetros análogos leído por el microcontrolador

Fuente: Elaboración Propia

3.2.2.2. Procesamiento:

Al realizar la configuración del puerto análogo para la lectura y el muestreo de los datos, es necesario la obtención de la trama análoga en base a los "*triggers*", generando las señales de Co-contracción, mantenido, doble pulso y triple pulso, estas siendo obtenidas de su forma pura análoga, y verificar que estos superan el umbral de voltaje DC de referencia.

Siendo estos verificables de forma gráfica y confirmar las diferentes tramas análogas.

a. Co-contracción (CC)

Esta ocurre cuando el voltaje generado supera el nivel de referencia en un periodo de tiempo corto predeterminado.

Figura 3.12: Valor de voltaje para OMG con referencia con co-contracción

Fuente: Elaboración propia

b. Mantener y soltar (HO)

Esta ocurre cuando el voltaje generado supera el nivel de referencia por un largo periodo de tiempo máximo predeterminado.

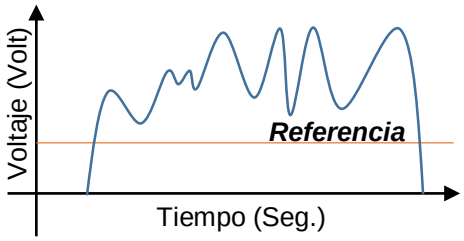


Figura 3.13: Valor de voltaje para OMG con referencia con Mantener-soltar

Fuente: Elaboración propia

c. Doble impulso (DI)

Esta lógica ocurre cuando el voltaje generado genera doble variación de voltaje, es decir que supere dos veces el nivel de referencia en un periodo de tiempo determinado.

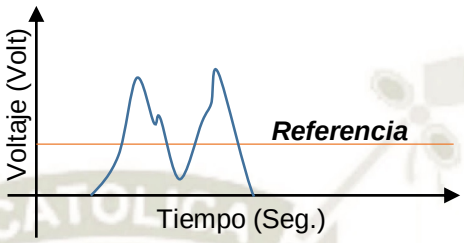


Figura 3.14: Valor de voltaje para OMG con referencia con doble impulso

Fuente: Elaboración propia

d. Triple impulso (TI)

Esta lógica ocurre cuando el voltaje generado genera doble variación de voltaje, es decir que supere tres veces el nivel de referencia en un periodo de tiempo determinado.

Figura 3.15: Valor de voltaje para OMG con referencia con triple impulso

Fuente: Elaboración propia

e. Doble mantener y soltar (DHO)

Esta lógica ocurre cuando el voltaje generado genera una co-contracción combinada con la lógica de mantener y en un periodo de tiempo determinado.

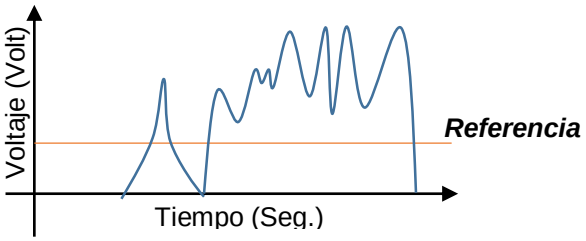


Figura 3.16: Valor de voltaje para OMG con referencia con doble impulso

Fuente: Elaboración propia

En la etapa de software se podrá obtener la información más detalladas como se programó en la sección 3.3.3 y como se desarrolló la lógica de control.

3.2.2.3. Clasificación:

En el procesamiento se busca ver las señales de forma digital, es decir verlos en valores de 1's y 0's, en mi caso utilice lógica directa, es decir cuando supere el umbral de referencia, este automáticamente es 1 y menor a ese será 0, se mantendrá siempre cuando cumpla la condición (Figura 3.17 a la figura 3.21).

a. Co-contracción (CC)

Luego de haber obtenido la señal de **co-contracción** análoga, esta es filtrada al cruzar el umbral de voltaje y convertida a valor digital.

Figura 3.17: Valor de voltaje para OMG con referencia con co-contracción

Fuente: Elaboración propia

b. Mantener y soltar (HO)

Luego de haber obtenido la señal de **mantener-soltar** análoga, esta es filtrada al cruzar el umbral de voltaje y convertida a valor digital, similar a lo anterior.

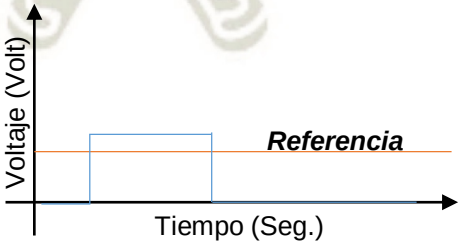


Figura 3.18: Valor de voltaje para OMG con referencia con mantener-soltar

Fuente: Elaboración propia

c. Doble impulso (DI)

Luego de haber obtenido la señal de doble impulso análoga, esta también será filtrada al cruzar el nivel de referencia y convertida a valores digitales.

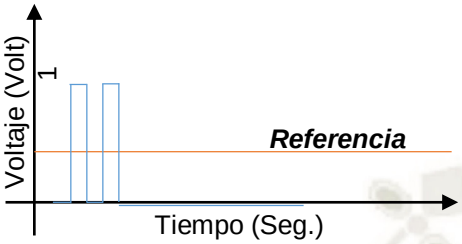


Figura 3.19: Valor de voltaje para OMG con referencia con doble impulso

Fuente: Elaboración propia

d. Triple impulso (TI)

Luego de haber obtenido la señal de triple impulso análoga, esta también será filtrada al cruzar el nivel de referencia y convertida a valores digitales.

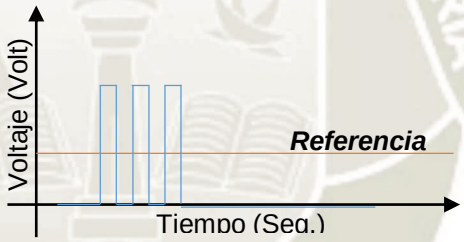


Figura 3.20: Valor de voltaje para OMG con referencia con triple impulso

Fuente: Elaboración propia

e. Doble mantener y soltar (DHO)

Esta señal es una mezcla de una co-contracción y simple impulso, esta también será filtrada al cruzar el nivel de referencia y convertida a valores digitales.

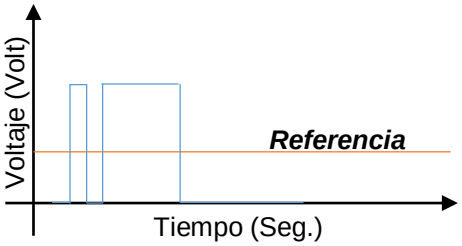


Figura 3.21: Valor de voltaje para OMG con referencia con DHO

Fuente: Elaboración propia

3.2.3. ETAPA 3 – Movimiento de los dedos

Ya ejecutada la etapa de procesamiento ahora se requiere que se aplique una lógico de control para que envíe señales de salida para que ejecute las diversas acciones siendo dependientes de la forma de control estas sean programados.

Para realizar esta tarea se requiere identificar cuántos actuadores son necesarios y se van a utilizar en el sistema.

Para el sistema mecánico de la mano se seleccionó el modelo de ***"EXIII HACKBERRY"*** siendo este detallado en la sección 3.4.2, este sistema requiere tres actuadores ordenados en la tabla N°3.

Tabla N° 3: Actuadores de la mano

POSICIÓN	DEDOS DE LA MANO	TIPO
Actuador 01	Pulgar	Servomotor
Actuador 02	Índice	Servomotor
Actuador 03	Meñique, anular, dedo medio	Servomotor

Fuente: Elaboración propia

Es decir, un actuador se enlaza con el dedo pulgar, el segundo se enlaza con el dedo índice y el ultimo actuador por un mecanismo enlaza el dedo menique, anular y el dedo medio, con la señal adquirida y procesada del sensor IR en el microcontrolador por un método de control de PWM se envía las señales de pulsos a los servomotores (Figura 3.22).

Figura 3.22: Clasificación de dedos de la mano

Fuente: Wikipedia, Dedos de la mano

3.2.4. ETAPA 4 – Interfaz Virtual LABVIEW

En esta etapa el objetivo es poder monitorear las diferentes variables sin tener dependencias directas es decir que el sistema mano no requiere necesariamente de la conexión al LABVIEW (Figura 3.23b) para su funcionamiento.

El protocolo de comunicación utilizado es serial, donde se envía y recibe la trama, donde se confirmar, almacenar, verificar y hacer el correcto DEBUGGER.

Para el paso inicial del DEBUGGER en necesario verificar las señales análogas y digitales de la forma visual es utilizando el puerto serial del microcontrolador y conectando al computador y utilizar la interfaz ***Energia*** (Figura 3.23a) y poder monitorear los datos seleccionados a la visualización, esta usa IDE Arduino, luego de seleccionar y verificar que esta envía correctamente la trama de datos, el siguiente paso es ya enviar esta data se al programa que se usara para el GUI que es el LABVIEW(Figura 3.23b) con el cual modificares y adaptaremos la data para poder visualizar y dar un correcto funcionamiento.

Toda esta etapa se desarrollará en la etapa de programación, en la sección 3.3.5 y se explicará paso a paso y se detallará el correcto funcionamiento.

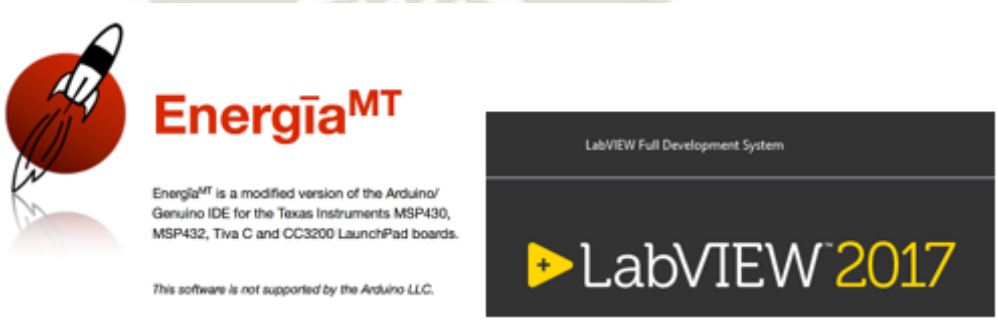


Figura 3.23: Softwares utilizados de (a) Energía.nu, (b) LABVIEW

Fuente: Captura de pantalla de la iniciación de cada programa

3.3. PROGRAMACIÓN

Para poder realizar la programación es necesario tener conocimiento de criterios para el desarrollo del programa, los cuales serán explicados en el siguiente la Figura 3.24 constando en cuatro partes.

Figura 3.24: Criterios para el desarrollo de la programación

Fuente: Elaboración propia

En los conocimientos de estructuras, variables, funciones esto va dirigido a los lenguajes de programación, ya que para el desarrollo de proyecto este conocimiento se da como principio básico, tanto en el software de LABVIEW y ARDUINO, a lo que me refiero al uso de estructuras cíclicas con la interacción de las funciones y las variables que el programador considere relevante.

Es muy importante que todas las variables actúen sin interrumpir a las unas a las otras y actúen teóricamente de manera simultánea, a esto siendo esto llamado el proceso de multitarea, también conocido en la cicla en inglés como “MULTITASKING”, con lo cual hacemos que muchas variables funcionen en

tiempo real, por ejemplo el encendido y apagado de leds en un periodo de 2 segundos en el cambio de cada flanco además de ejecutar la tarea de leer las variables del sensor y enviar las diferentes señales, sin que un proceso interrumpa a otro a esto se llama un proceso de multitarea, que estará presente en este sistema.

Para poder optimizar y se mantenga un orden para mejorar o modificación para futuras versiones se utilizó máquinas de estado finito ("FINITE STATE MACHINE") para que las variables externas realizan la transición de estados, también nos va a ayudar a realizar el código más simple y fácil para corrección de errores en código o de mejoras de este, debido a su estructura secuencial que ayuda a cambiar de estado según la señal que seleccionemos y el usuario envíe la señal de control.

Los sistemas matriciales y vectoriales son unas de mis últimas consideraciones básicas donde podemos manejar y almacenar múltiples variables solo variando y modificación la posición como la simplificación de código, sin que tener que repetir la misma tarea para otra variable. Para poder realizar se requiere entender cómo lograr realizar un código universal y simplificar el sistema en sistemas de tipo matricial como el vectorial.

3.3.1. BASES DE PROGRAMACIÓN

3.3.1.1. Programación ARDUINO (ENERGIA IDE)

La programación está organizada en 4 secciones principales, las cuales gran parte del código lo redacte con variables en ingles por facilidad de escritura, ya que me es más fácil escribir palabras en inglés y estas siendo más cortas que nuestra lengua materna por ejemplo estructura como "***STRUCTURE***" pero los comentarios serán redactados en español para mayor compresión del usuario o personas que deseen revisar este documento y las 4 secciones son las siguientes:

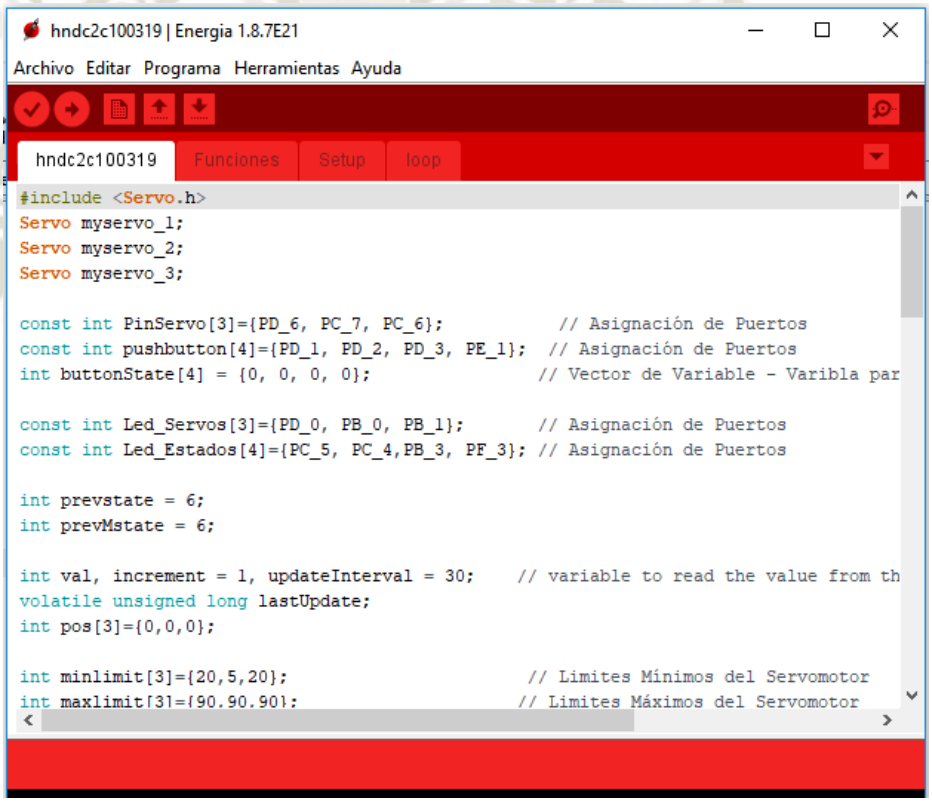


Figura 3.25: ENERGIA IDE PROGRAMACIÓN

Fuente: Elaboración propia

a) **Handc**: Definición de variables globales, contantes, matrices, vectores y condiciones iniciales.

Figura 3.26: Definición de variable globales, constantes y librerías

Fuente: Elaboración propia

b) **Funciones**: Se crea la función "**servoControl"** con la cual se controla y se limita el control de los servomotores instalados en el sistema para este sistema siendo 3, también se tiene "**writeField**" donde se transforma la variable ARDUINO para ser enviada a través del puerto serial al programa LABVIEW.

Figura 3.27: Apartado de funciones

Fuente: Elaboración propia

c) **Setup**: Aquí se configura la señal de comunicación serial para el monitoreo en la cual enviara todos los parámetros al LABVIEW, ya siendo este mi caso, teniendo en cuenta que este no depende del LABVIEW, es decir, puede funcionar de forma autónoma.

Se configuran los puertos de entrada y salida del sistema de control, también se setea los tiempos de los multi-pulsos, esto es para las co-contracciones, mantener y soltar, doble pulsos, triple pulsos y una que es una mezcla de la co-contracción con el mantener y soltar.

Finalmente, se setea el comando de calibración para el sensor IR en la detección en sus rangos máximos y mínimos de muestreo para luego en el "*loop*" ser escalado en un rango porcentual de lectura que va del 0% al 100%.

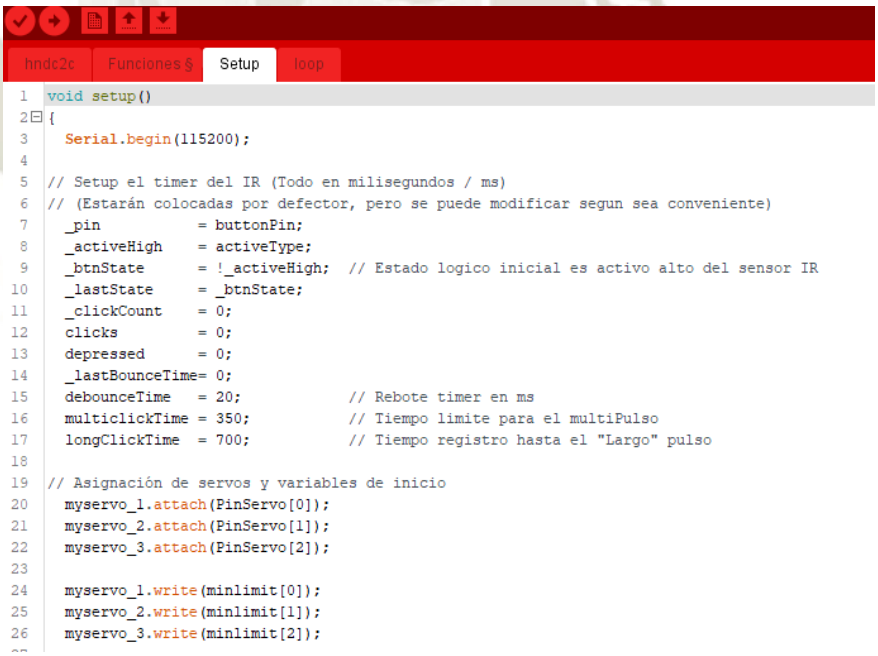


Figura 3.28: Configuración de puertos y calibración.

Fuente: Elaboración propia

d) **Loop**: En esta parte se desarrolla la programación principal del sistema y empezamos primero con la adquisición de la variable análoga principal de sistema siendo acondicionada, procesado y dando lógica

FSM para funcionamiento del sistema actual y utilizar comandos "*millis"* para la realización de la multitarea.

Finalmente, luego de ser clasificada y codificada para enviar a los actuadores las señales de control y realizar el desplazamiento del sistema mecánico-electrónico a demás presentaremos una parte del código que se encarga de enviar los parámetros a través del puerto serial al computador para luego ser leídas y visualizadas en el programa de LABVIEW.

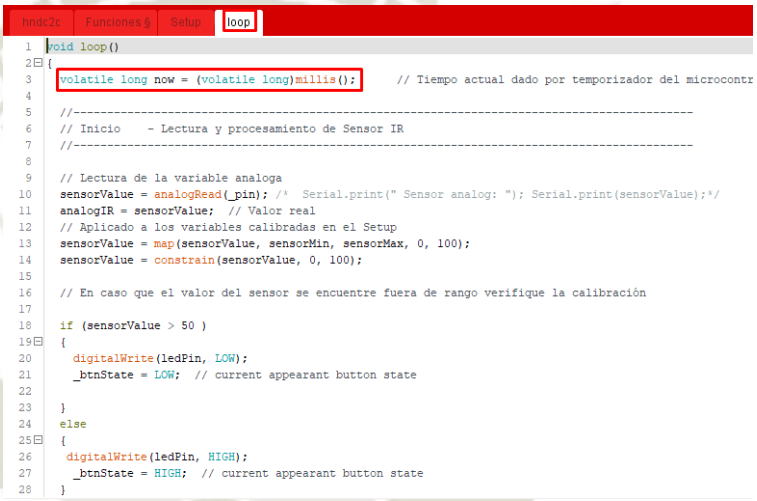


Figura 3.29: Bucle loop

Fuente: Elaboración propia

Toda la codificación en IDE ARDUINO se podrá revisarlo en el archivo anexo, sólo se indexará en esta etapa los diagramas de flujo de cada parte de código que se considere importante, los cuales serán explicados paso a paso del comportamiento y su desarrollo.

3.3.1.2. Programa LABVIEW

Para la transmisión serial se necesita la aplicación VISA de LABVIEW, conocer el VI, donde realizaremos la programación que recibirá la data enviada del microcontrolador, al lograr corroborar el correcto funcionamiento se realizar la compilación del proyecto del LABVIEW y realizar un archivo ejecutable (EXE) para la interacción con el usuario y poder proteger la programación.

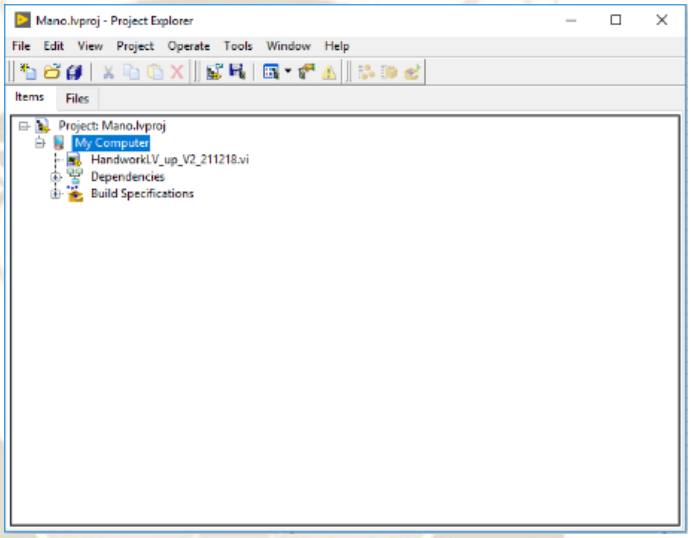


Figura 3.30: Proyecto LABVIEW ***mano.lvproj***

Fuente: Elaboración propia

Primero se desarrolló la creación del proyecto con el nombre de ***"mano.lvproj"*** **(**Figura 3.30**)**, donde se guardará la interfaz visual del LABVIEW creada y en donde se realizará la compilación del programa desarrollado, luego se creará la interfaz visual requerida con su respectiva programación para su ejecución (Figura 3.31).

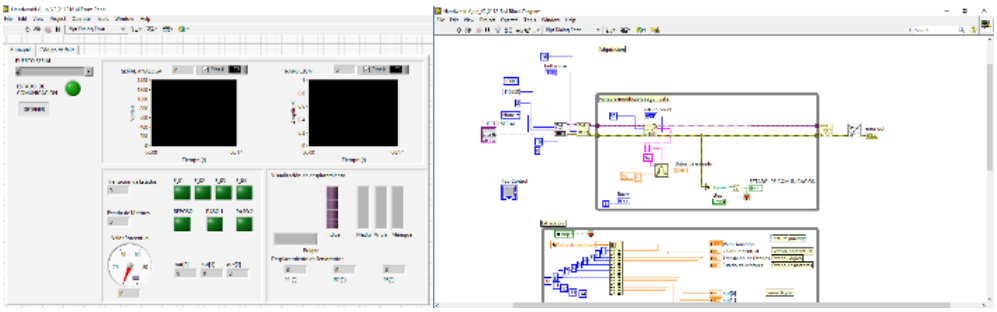


Figura 3.31: VI LABVIEW

Fuente: Elaboración propia

Finalmente, para no tener la necesidad del uso del programa de LABVIEW se prosiguió a la creación de un archivo ejecutable en el LABVIEW PROYECT (Figura 3.32) mencionado anteriormente y serán explicados más detalladamente en 3.3.5 .

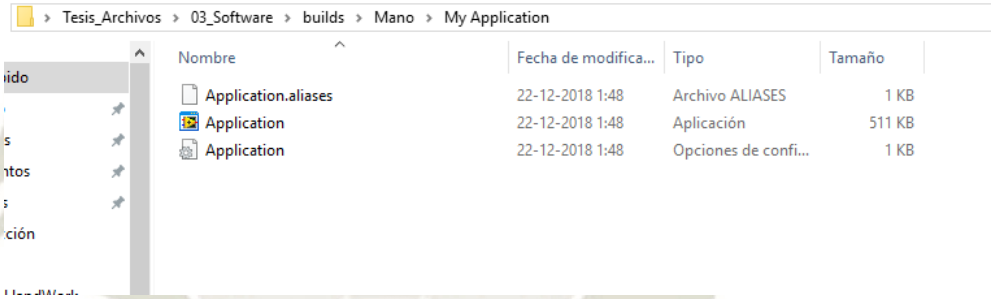


Figura 3.32: Ejecutable del proyecto

Fuente: Elaboración propia

3.3.2. ETAPA 1 – Contracción muscular

Se configura la lectura análoga al microcontrolador, se hace el arreglo al sensor para la adquisición de la señal, procesamiento de la señal y está siendo preparada para el procesamiento.

Esta etapa se ha va dividir en tres partes explicadas a continuación se explicarán detalladamente:

Primero requerimos que el microcontrolador lea un rango adecuado de voltaje, para lograrlos es necesario que el módulo IR sea posicionado en el brazo con la banda de forma adecuada.

a. La primera es conocida como **calibración** se detecta los rangos máximos y mínimos donde llega el sensor que van a ingresar al microcontrolador, ya que la forma, posición y tipo de piel afecta el voltaje de salida, siendo esta tarea la más importantes.

   Esta función de se cumple en el SETUP, primero se definen las variables locales y globales que actúan. Luego se usa un bucle “while” con la condición que ejecute durante los primeros 5 segundos para que salga del bucle.

   En este bucle se realiza la lectura análoga por el puerto análogo seleccionado y esta almacenada en la variable “sensor” además de actualizar el tiempo actual durante cada iteración.

   Para que funcione el sistema de calibración hay presento dos condiciones “if” donde en la primera condición establecemos que el valor máximo en 0 con el que empieza esta variable del sensor siendo “sensorMax=0”, en la condición “if” se compara el valor leído con este valor, cumpliendo con la condición si la lectura del sensor es mayor el sensorMax ingresa al cumplir la condición, luego almacenando esta nuevo valor en la variable sensorMax, siempre que

cumpla la condición, ya tenemos los valores máximos. El funcionamiento es similar para la otra condición "if" pero con el valor "sensorMin=1023" que empieza con un valor alto y comparando encuentra un valor menor que lo almacena "sensorMin". Al finalizar este bucle "while" luego que transcurra el tiempo obtendremos nuestro rango real de operación.

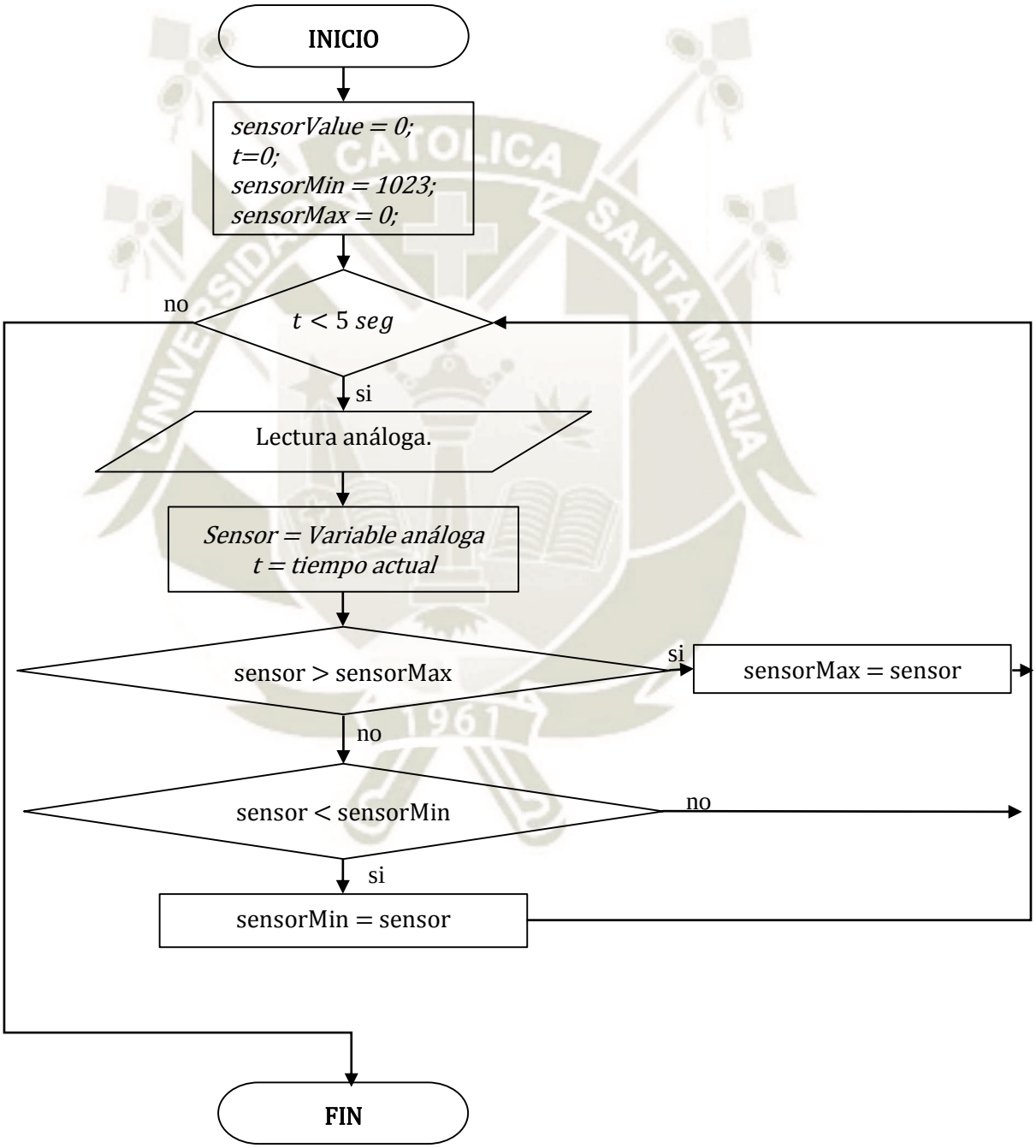


Figura 3.33: Diagrama de flujo de calibración

Fuente: Elaboración propia

b. En la segunda parte se realiza el **mapeo**, con esto le damos un cambio de escala para el sistema, ya sabiendo los rangos máximos y mínimos de los valores analógicos que ingresan al microcontrolador, nos enfocamos en el cambio de escala a valores porcentuales que desplazarían de 0% a 100% el valor analógico.

$$out = \frac{(in - in_{min}) * (out_{max} - out_{min})}{(in_{max} - in_{min}) + out_{min}} \qquad \textit{Ecuación 3.1.}$$

Parámetros

in: El valor análogo para el escalado

in_{min}: El valor mínimo del actual rango

in_{max}: El valor máximo del actual rango

out_{min}: El valor máximo del nuevo rango

out_{max}: El valor máximo del nuevo rango

Gráficamente podemos ver la variación del valor analógico y realizado el escalado a valores porcentuales dados en el mapeo en la figura 3.34.

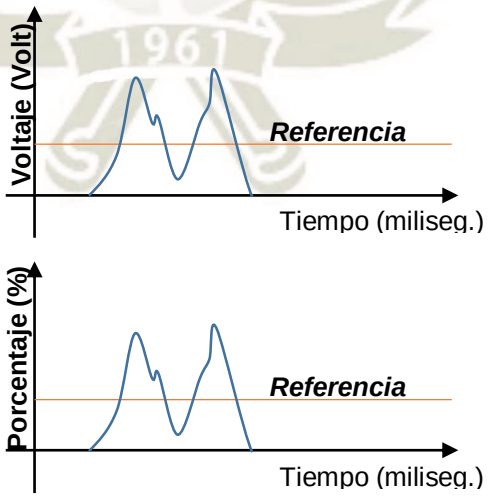


Figura 3.34: Conversión del valor

Fuente: Elaboración propia

Esta parte del código se ejecuta en el "LOOP" siendo una estructura que repite siempre. En el diagrama de flujo primero definimos locas, globales luego leemos el valor análogo adquirido por el microcontrolador en tiempo real, siguiendo aplicamos la fórmula de mapeo (*Ecuación 3.1.*) y finalmente el valor obtenido al aplicar la formula se guarda en la variable "sensorValue", con esta parte ya tenemos los datos acondicionados para trabajar de forma más practica debido que desde ya los valores van de 0% a 100% mencionados anteriormente.

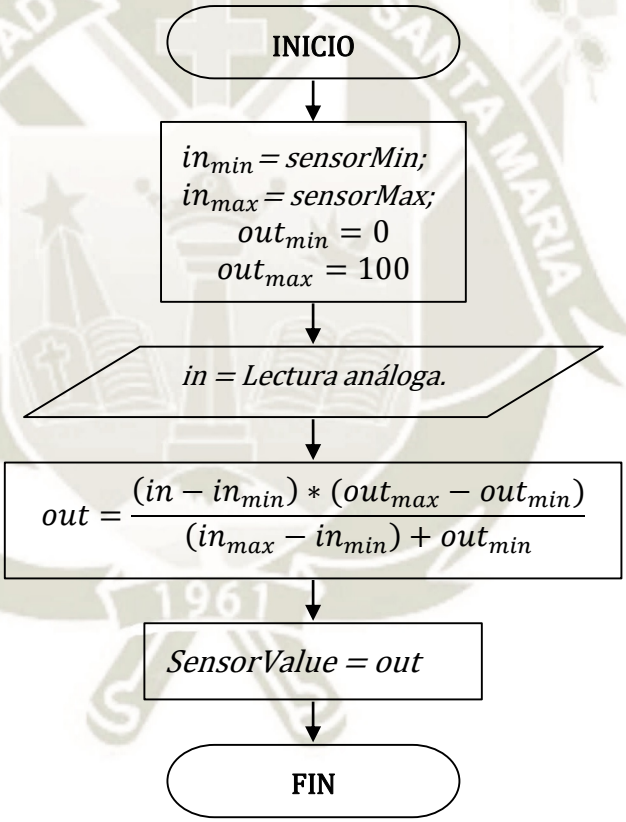


Figura 3.35: Diagrama de flujo de escalado (Mapeo)

Fuente: Elaboración propia

c. Finamente **conversión a lógica booleana (**figura 3.36**)**, donde tiene que vencer el umbral de salida para obtener una salida lógica de 1 y 0.

En nuestro diagrama de flujo tenemos como variable de entrada como el "sensorValue", está accediendo siempre a una condición "if" siempre que

cumpla la condición que el valor porcentual sea mayor al 50% en valor sensado, está activando una señal digital al cumplir la condición de btnState=LOW y al no cumplir enviando btnState=HIGH. Con esta conversión finalizamos la adquisición digital.

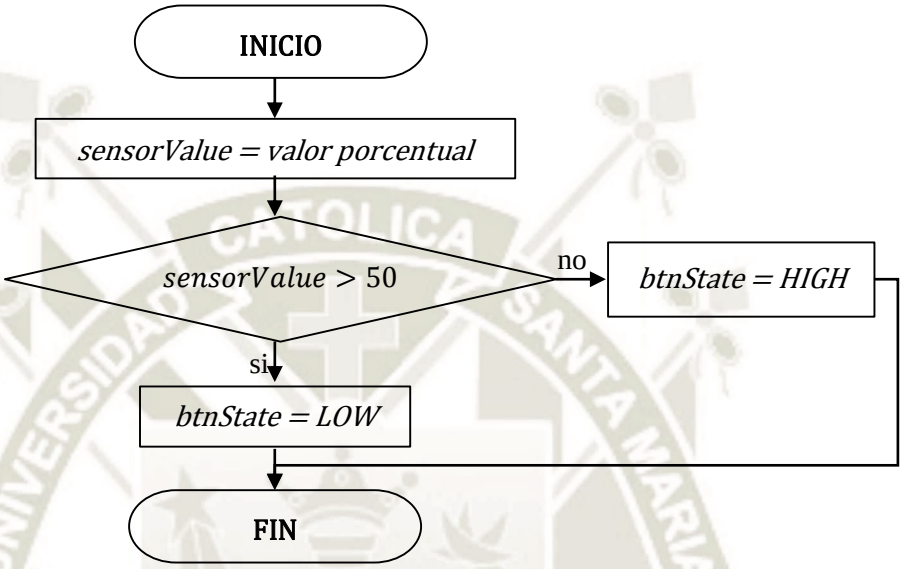


Figura 3.36: Diagrama de flujo de conversión booleana

Fuente: Elaboración propia

De forma gráfica (Figura 3.37) se puede visualizar en la figura siguiente, encontrando la transición a los valores booleanos al superar el límite para detección.

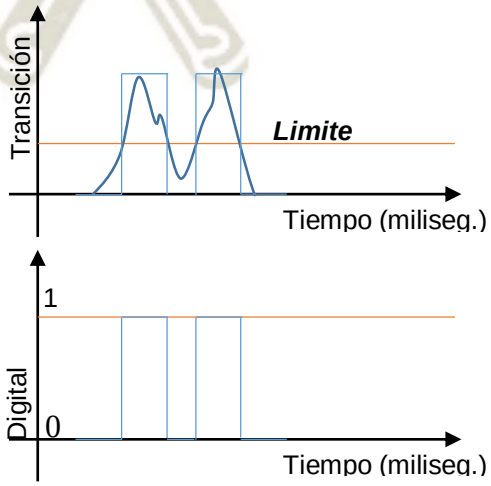


Figura 3.37: Explicación grafica del diagrama de flujo

Fuente: Elaboración propia

3.3.3. ETAPA 2 – Procesamiento de Señal

Durante el desarrollo de esta etapa se realizará la lectura análoga al microcontrolador y posteriormente realizar el procesamiento para obtener la señal real de control. Debido a que manejamos una sola señal de entrada para realizar muchos procesos es necesario el uso de la multitarea dando una importancia al uso de las funciones millis() o micros() el uso de las interrupciones. Para este proyecto se usó la función millis() en el la realización y cumplimiento de las tareas.

Con los cálculos realizados en la aparta anterior y el dado acondicionamiento, luego realizando la tarea usar los valores porcentuales y convertirlos a valores digitales, es decir, en valores booleanos.

En esta etapa se basa en condición "if" con comparación en la temporización de estos y siempre cuando cumpla la condición de tiempo accederá a estos.

Al finalizar esta etapa de procesamiento obtenemos valores digitales de SIMPLEPULSO, DOBLE PULSO, TRIPLE PULSO, pulso mantenido entre otros, siendo estas variables las más importantes para el funcionamiento de la lógica de control.

Para este apartado se realizarán 2 diagramas de flujos que resumen el comportamiento base de este apartado:

a. Detección de tipos de pulsos

b. Selección de tipos de pulsos

a. Detección de tipos de pulsos

Está basado en la detección del tipo de pulso además de la detección del ruido y filtrar la señal útil (figura 3.38). En resumen, inicialmente nos permite detectar un pulso correcto y un pulso largo durante un periodo de tiempo establecido, al superar este tiempo las variables se reinician, como si no hubiera detectado la señal hasta que lo realice de forma correcta el usuario.

Una de señales principales la detección del ruido o para nuestro caso es conocido como tiempo de rebote "*debounceTime*" el que establecí con un periodo de 20 milisegundos, también tenemos el tiempo para la detección de pulsos o multipulsos es de 350 milisegundos y el ultimo es la selección para la lectura del pulso largo que su periodo máximo es de 700 milisegundos.

Estos valores son ajustados en base a prueba y error con el usuario que utilice el sistema de sensado debido a que cada usuario presenta su propia velocidad de respuesta.

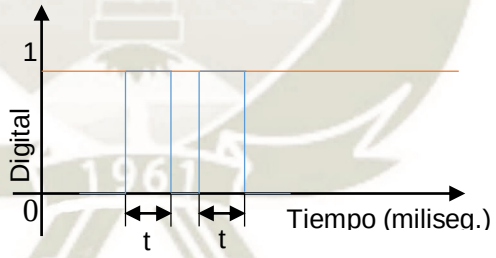


Figura 3.38: Establecimiento del periodo máximo.

Fuente: Elaboración propia

En la etapa uno ya podemos obtener los valores digitales en esta etapa nos aseguraremos que cumpla las condiciones en caso de no cumplir, simplemente se ignorar la señal que no cumpla las condiciones debido a su falta de claridad para el código. Dejemos como ejemplo el de doble pulso este no tiene que superar el tiempo establecido (figura 3.38), ya que podrá interpretar otro tipo

de señal y el de mantener-soltar tiene que superar un valor mínimo para detección de esta señal (figura 3.39).

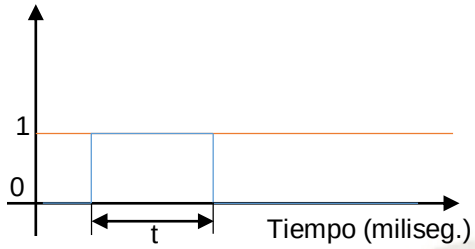


Figura 3.39: Valor de voltaje para OMG con referencia con mantener-soltar

Fuente: Elaboración propia

Proseguiremos con la explicación del diagrama de flujo (figura 3.40.), el cual resume de forma práctica el comportamiento para realizar la detección del tipo de pulso.

Inicialmente establecemos los periodos para poder detectar dichas señales. En la primera condición "if" nos preguntamos si hubo un cambio digital es decir de 1 a 0, ya que manejamos lógica inversa en la detección de pulsos, si hay cambio de estado envía confirmación de cambio de estado, en la segunda condición "if" corroboramos si encontramos ruido, en caso de encontrarlo se mantiene la condición. En la tercera condición "if" confirmamos que el pulso se mantiene estable, si no lo es, mantiene la condición del estado, al caso de cumplir la condición verificamos si hay cambio de estado y al cumplir esta entrega la cuenta del primer pulso. La cuarto condición "if" se encarga de verificar si realizamos más pulsos y encargarlos de contarlos. Y, por último, esta condición da un intervalo de tiempo mínimo, para nuestro proyecto se seleccionó el valor 700 milisegundos que al superarlo cuenta como pulso largo.

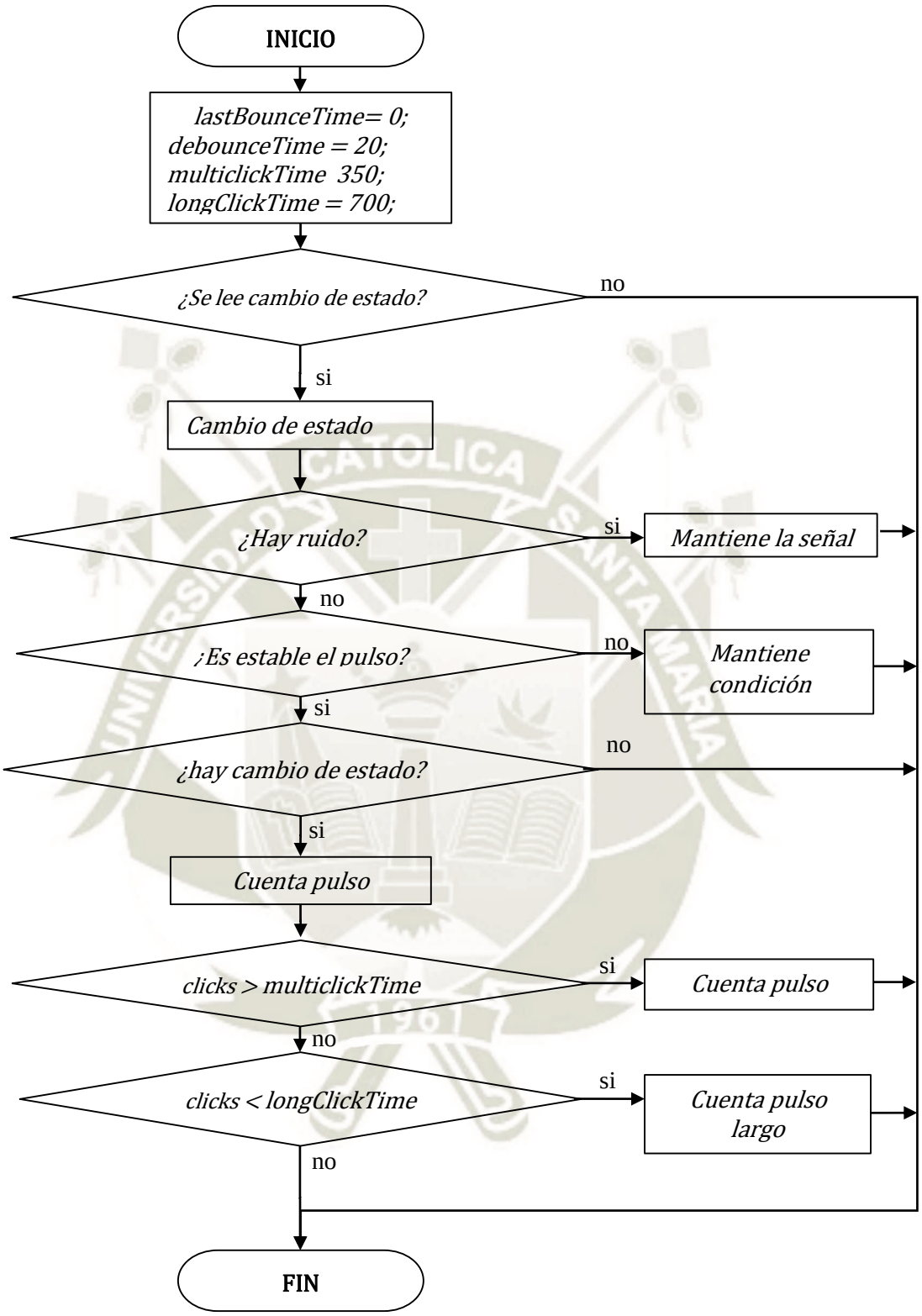


Figura 3.40: Diagrama de flujo de detección del tipo de pulso

Fuente: Elaboración propia

b. Selección de tipos de pulsos

Ya siendo posible detectar el tipo de pulso y filtrar el ruido presente, es tiempo de seleccionar los pulsos que nos sirve para poder controlar el proyecto.

Como variable de entrada obtenemos la lectura de pulsos que pasa a través de un máximo de seis condiciones "if" para seleccionar el tipo de señal. En la primera condición "if" ya detectamos un pulso en el periodo de tiempo establecido nos da como respuesta correcta la co-contracción (CC).

En la segunda condición confirmamos doble pulso, si cumple la condición obtendremos la señal de doble impulso (DI) y en el caso de cumplir para a la siguiente condición.

En la tercera condición confirmamos triple impulso, al cumplir la condición obtendremos la señal de triple impulso (TI) y al no cumplir pasamos a la siguiente condición.

En la cuarta condición verificamos si la señal no se actualizó, esta guarda una señal conocida sin efecto para el control, al no cumplir pasa a la siguiente condición.

En la quinta condición confirmamos la condición mantener soltar (HO), al cumplir como las anteriores se almacena, al no cumplir pasa a la siguiente condición.

En la última condición verificamos el de doble pulso largo (DO), al cumplir con lo anterior esta variable también se almacena, y al no cumplir para al fin.

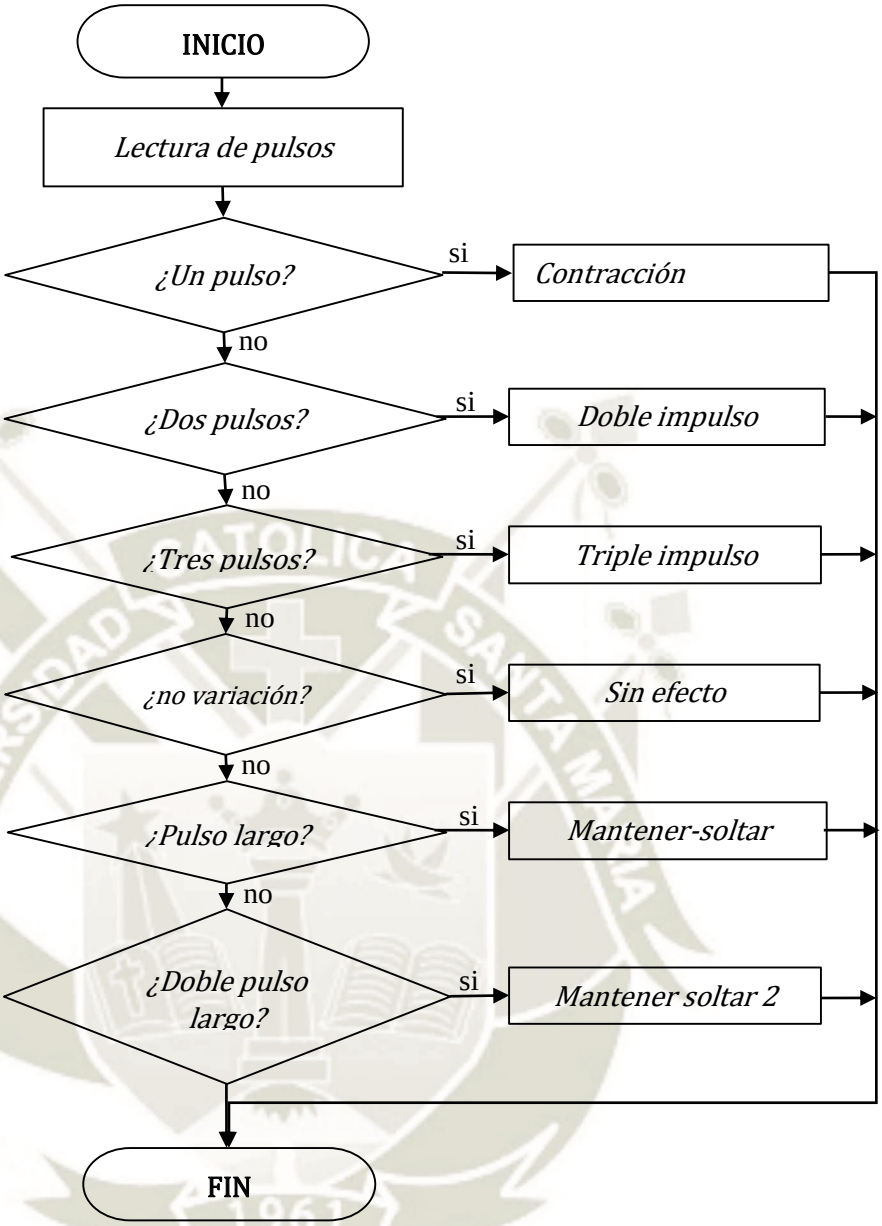


Figura 3.41: Diagrama de flujo de selección los tipos de pulsos

Fuente: Elaboración propia

3.3.4. ETAPA 3 – Movimiento de los dedos

Para el control de hardware se utilizó la librería de ARDUINO "**Servo.h"** que viene preconfigurada con la frecuencia fija y poder variar el ancho de pulso necesario para el desplazamiento angular que presenta cada servo motor, teóricamente se encuentra dentro de un rango de 0 por ciento a 100 por ciento.

En esta etapa se utiliza los servomotores para poder realizar el desplazamiento de los dedos, para que actúen estos, se requiere que el microcontrolador procese la información anterior por alguna lógica que integre las etapas anteriores con esta, que se encarga de realizar la ejecución del control de los actuadores y la lectura del sensor y así poder realizar el procesamiento en conjunto.

Para poder mejorar y reducción del código de procesamiento requerí mejorar la lógica y su uso respectivo de estado debido al punto positivo de trabajar con valores binarios.

Por lo que al tener 3 servomotores dan como resultado 8 condiciones posibles de desplazamiento.

$$2^3 = 8$$

Servomotor 1: Pulgar

Servomotor 2: Medio anular, meñique.

Servomotor 3: Índice

En base a lo anterior se realiza la tabla de estados ordenándolos de 0 a 7 siendo la cantidad de 8 estados, hay que tener en cuenta que 1 es abierto y 0 es cerrado idealmente.

El primer bit es el servomotor 3, el segundo es el servomotor 2 y el tercero es el servomotor 3 como se ve en la tabla siguiente.

Tabla N° 4: Diagrama de estados ideal y real

	ESTADO IDEAL			ESTADO REAL		
ORDEN	**S1**	**S2**	**S3**	**S1**	**S2**	**S3**
00	0	0	0	0	**1**	0
01	0	0	1	0	**1**	1
02	0	1	0	0	**0**	0
03	0	1	1	0	**0**	1
04	1	0	0	1	**1**	0
05	1	0	1	1	**1**	1
06	1	1	0	1	**0**	0
07	1	1	1	1	**0**	1

Fuente: Elaboración propia

S1: Servomotor 1

S2: Servomotor 2. En el estado real el valor booleano se invierte.

S3: Servomotor 3

Para la lógica se decidió utilizar diagramas de estados, los cuales fueron divididos en 4 estados, es decir que en la tabla N°5 se decidió realizar 2 tareas por estados, es decir, internamente requiere condiciones adicionales para realizar la transición interna y los datos de transición externa.

Tabla N° 5: Tabla de transiciones internas ideales

	Ideal Seleccionado			Real		
Estado 0	110	↔	011	100	↔	001
Estado 1	000	↔	111	010	↔	101
Estado 2	100	↔	101	110	↔	111
Estado 3	001	↔	010	011	↔	000

Fuente: Elaboración propia

En las sub-transiciones que son cambios internos dados en cada estado, estos reflejan de forma visual la apertura o cierre del actuador respectivamente para cada estado.

Para la transición de estados se usa los “triggers” como condiciones, siendo estas generadas inicialmente por el usuario.

a. Lógica principal:

Se encuentra dividido en cuatro estados, desarrollado y creados de esta forma por mi propia autoría para la lógica de control, además de presentar tres condiciones para que se realice la interacción de un estado a otro en el diagrama de estados.

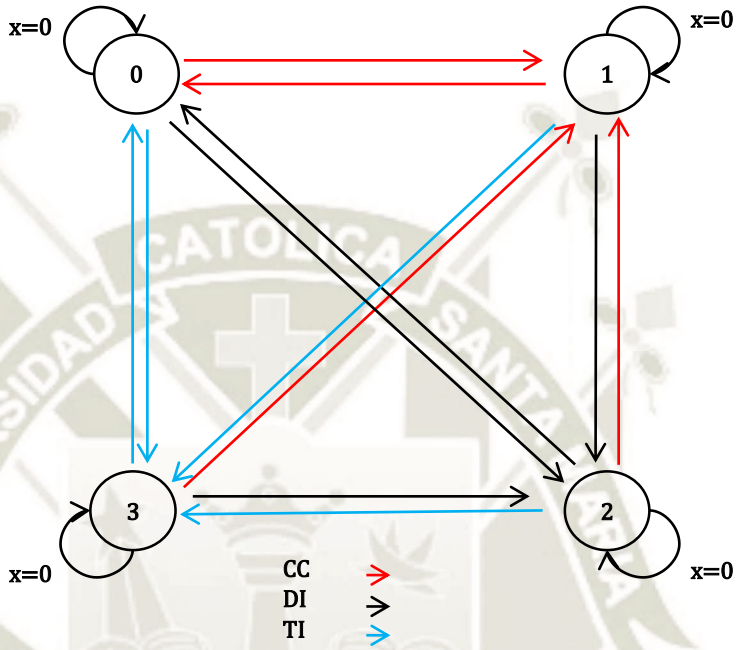


Figura 3.42: Diagrama de estados para transición de es estados

Fuente: Elaboración propia

Las condiciones de transición son:

1. CC: Co-contracción: Es la condición principal para la transición del estado 0 hacia el estado 1.
2. DI: Doble impulso: Es la condición principal para la transición de los demás estados hacia el estado 2.
3. TI: Triple impulso: Es la condición principal para la transición de los otros estados hacia el estado 3.

En la realización de la tabla que se realizó (tabla N°6), donde se puede explicar el funcionamiento y de forma más simple, por ejemplo; si no s encontramos en el

estado 0 y deseamos movernos al estado 3, debemos de generar la señal de triple impulso para trasladarnos a este estado, y así hacia los demás estados.

Tabla N° 6: Tabla de Estados con selección de disparadores

IN \ OUT	0	1	2	3
0	X	CC	DI	TI
1	CC	X	DI	TI
2	DI	CC	X	TI
3	TI	CC	DI	X

Fuente: Elaboración propia

b. Lógica de transiciones internas

Para poder realizar la transición establecido en la tabla N°5 se requieren condiciones adicionales de cambio interno de estado, los cuales fueron seleccionados en base a mi criterio son los siguientes:

1. HO: Mantener-abrir, para generar esta señal es necesario tener contra ido el músculo.
2. DHO: Doble mantener y abrir: Siendo una modificación del CC (Co-contracción) y el HO (Mantener-Soltar), para su ejecución se requiere una co-contracción y luego realizar acción de contraer y soltar.

El estado de motor sirve para ayudar a la reducción de código y la complejidad de su programación, este se encarga del cambio de transición interna en cada estado, siendo básicamente divido en dos estados interiores con los cuales activamos y desactivamos los servomotores.

Por lógica de funcionamiento de los motores y servomotores se desplazan tanto en sentido horario, anti horario como el estado de reposo (Figura 3.43).

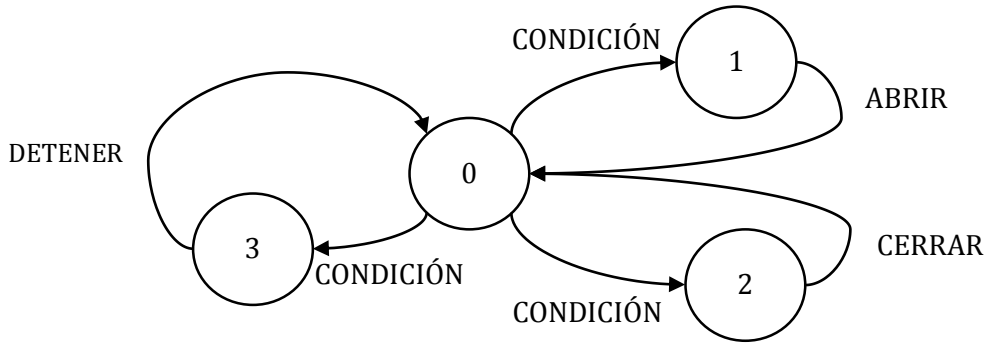


Figura 3.43: Diagrama de estados de lógica motores

Fuente: Elaboración propia

Las condiciones de los motores se dividen en tres simples sentencias abrir, cerrar y detener, que serán explicada a continuación y resumidas en la tabla N°7.

La ***"Acción 1"*** se encarga de desplazar el actuador de tal forma que visiblemente el dedo o conjunto de dedos emulen la condición de abrir.

La ***"Acción 2"*** se encarga de desplazar el actuador de tal forma que visiblemente el dedo o conjunto de dedos emulen la condición de cerrar.

La ***"Acción 3"*** se encarga de desplazar el actuador de tal forma que visiblemente el dedo o conjunto de dedos dejen de desplazarse.

Tabla N° 7: Lógica de actuación de motores

ESTADO	LÓGICA	ACCIÓN
Acción 1	0	Abrir
Acción 2	1	Cerrar
Acción 3	2	Detener

Fuente: Elaboración propia

Explicación de función de motores:

Luego del inicio tenemos como variables de entradas tres señales de control, una por cada servomotor dando un total de 3, y una variable de tiempo que se actualiza en el bucle principal LOOP.

El bucle "while" solo se ejecuta cada intervalo de tiempo, siendo este cada 20 milisegundo. Al ingresar al bucle se a ejecutar un bucle "for" con un conteo del 1 al 3, esto se debe al ser 3 servomotores, dando que cada ciclo para para un servomotor, dentro de este bucle inicialmente el tiempo se actualiza e internamente posee tres condiciones "if" para cumplir la lógica de incremento, decremento y mantener posición, además de dos condiciones "if" para limitar el desplazamiento.

En la primera condición "if" se compara con el valor numérico 0 y al cumplir hace la posición actual del servomotor incremente, al no cumplir pasa a la siguiente condición.

En la segunda condición "if" se compara con el valor numérico 1 y al cumplir hace la posición actual del servomotor disminuya, al no cumplir pasa a la siguiente condición.

En la tercera condición "if" se compara con el valor numérico 2 y al cumplir hace la posición actual del servomotor se mantenga y no ejecute ningún desplazamiento, al no cumplir pasa a la siguiente condición.

En la cuarta condición "if" la considero como una condición filtro y limitador de saturación para poder proteger al servomotor no se salga de sus valores mínimos y no dañar el hardware mecánico.

En la última condición “if” también la considero como una condición filtro y limitador de saturación para poder proteger al servomotor no se salga de sus valores máximos de desplazamiento y no dañar el hardware mecánico.

En resumen,

Es una función que se encarga de convertir las señales lógicas en valores que son legibles para los servomotores, del rango de 0 a 90 teóricamente (Figura 3.44).

Pero los valores reales máximos y mínimos los de servomotores son:

Para el primer servomotor su desplazamiento va en un rango de 20 a 90 grados teóricos.

Para el segundo servomotor su desplazamiento va en un rango de 5 a 90 grados teóricos.

Para el tercer servomotor su desplazamiento va en un rango de 20 a 90 grados teóricos.

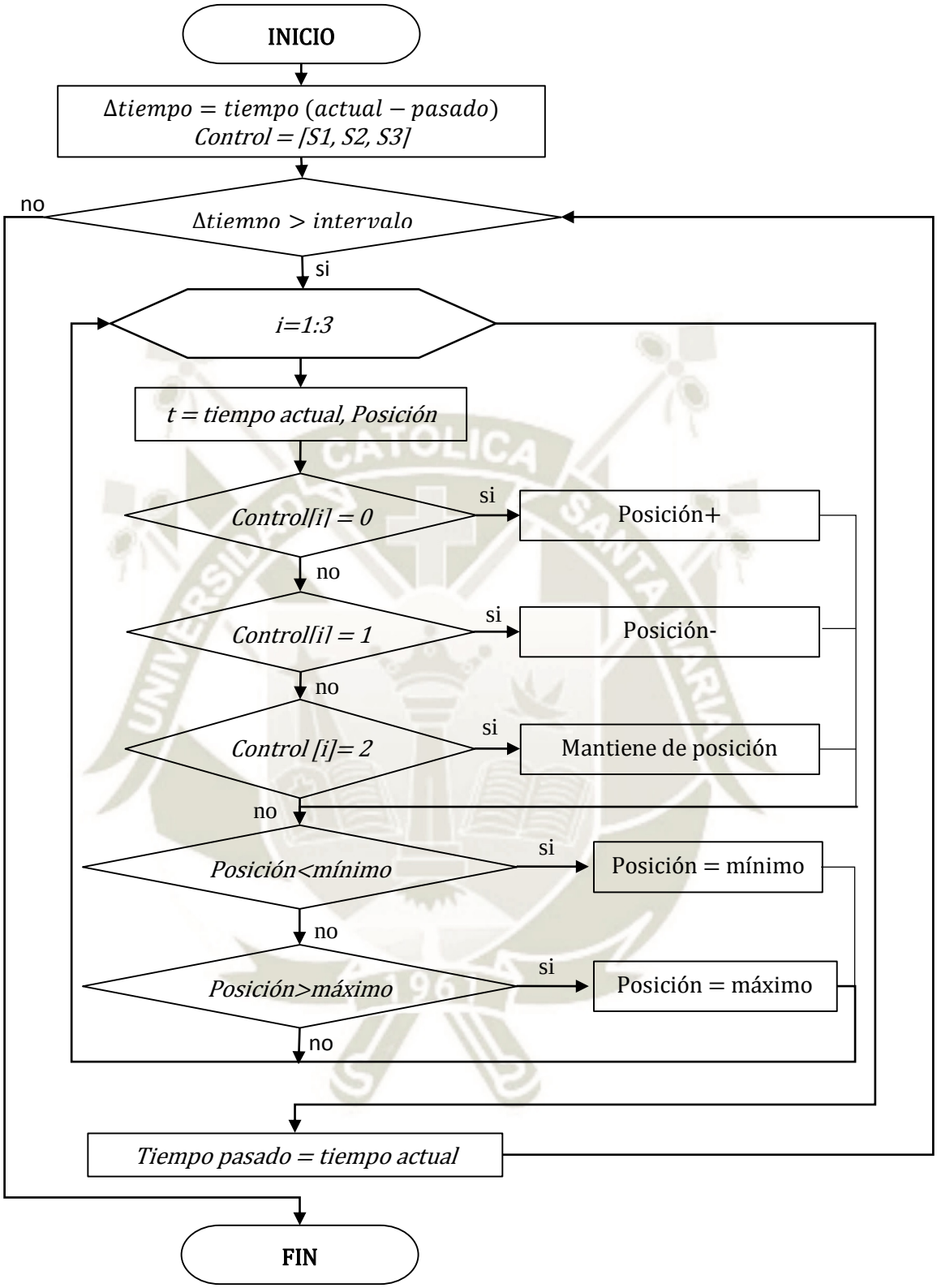


Figura 3.44: Diagrama de flujo para desplazamiento de los motores

Fuente: Elaboración propia

3.3.5. ETAPA 4 – Interfaz Virtual LABVIEW

En esta etapa se realiza el monitoreo multivariable en el LABVIEW sin que este tenga dependencia alguna al microcontrolador, solo su tarea es monitorear los parámetros si este se conecta vía serial, ya que este parámetro considero opcional para el usuario.

Para que se desarrolle de esta forma se requiere hacer un programa de envío y recepción de datos.

Siendo la secuencia del microcontrolador al LABVIEW por lo que se requiere un programa en cada interfaz.

La información que ingresa al LABVIEW es de forma vectorial cuyo parámetro de separación es la coma que viene con la trama de datos enviados y esta es enviada cada 5 milisegundos.

El valor que configuré de entrada es la letra I y el de finalización del vector es el salto de línea con lo cual detectamos que llega otra trama de datos.

Para la IDE ARDUINO se configura la frecuencia a 115200 baudios y se crea la función **writeField**() que convierte cualquier valor que se desea enviar a formato de texto conocido como "***STRING***" y separar por una coma la cantidad de datos que deseamos enviar.

En el LOOP se envía todas las variables que deseemos visualizar en el LABVIEW.

Antes de crear la interfaz virtual (VI) se crea el proyecto, el cual lo nombre con "***Mano.lvproj***", en donde se ordenará y dará un orden al proyecto de LABVIEW.

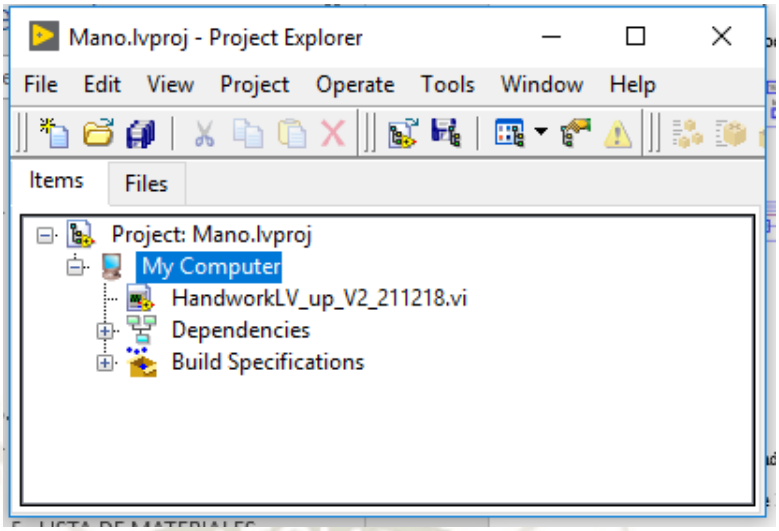


Figura 3.45: Proyecto Mano

Fuente: Elaboración propia

Para el LABVIEW se divide en dos partes:

1. Configuración de la señal de entrada.
2. Selección de señales para visualizar.
3. Preparación de la interfaz para visualización.

En la configuración de la señal de entrada se requiere la aplicación VISA de LABVIEW (Figura 3.46) para poder configurar y seleccionar los parámetros a muestrear y toda la trama adquirida enviarla a un vector para luego ser clasificado.

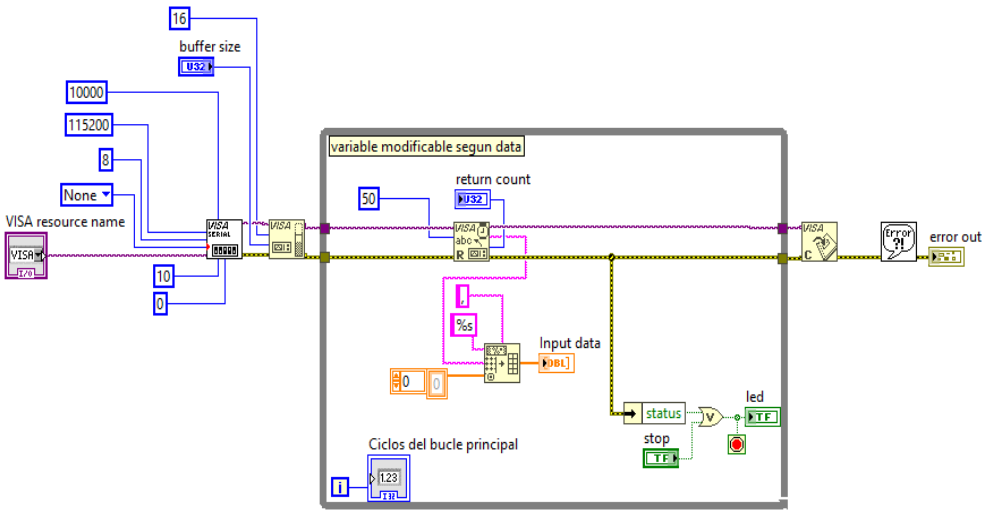


Figura 3.46: Adquisición por comunicación serial

Fuente: Elaboración propia

El vector donde se recibió los datos al VI del LABVIEW, la siguiente tarea es la extracción de los datos en forma individual (Figura 3.47), donde esta se ordena para que el programador lo pueda ver y clasificar estos parámetros, esta tarea es sencilla separando en su respectiva forma de visualización, todo esto separado en un nuevo bucle "*while*" para esta nueva tarea.

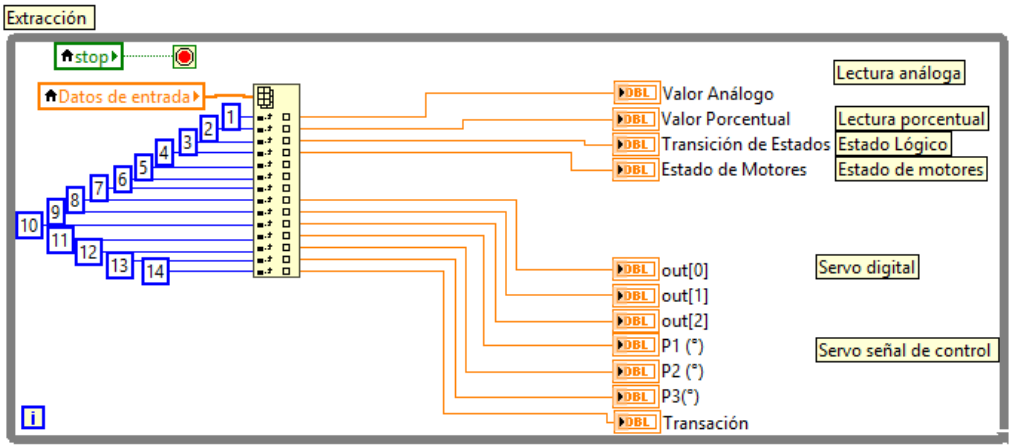


Figura 3.47: Separación de datos

Fuente: Elaboración propia

Luego de seleccionar los datos se separan en otro bucle donde los seleccionados clasifican para su visualización según sea requerido y como se desee mostrar (Figura 3.48).

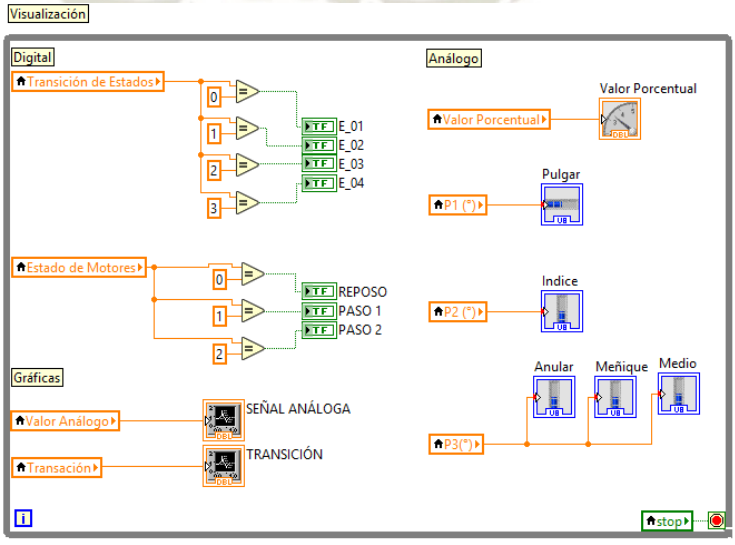


Figura 3.48: Selección de Iteración grafica para el usuario

Fuente: Elaboración propia

Finalmente se requiere una interfaz gráfica de usuario (Figura 3.49) con la se realizará monitoreo de las variables seleccionadas siendo las más importante el comportamiento de la señal IR como las señales de salida a los servomotores.

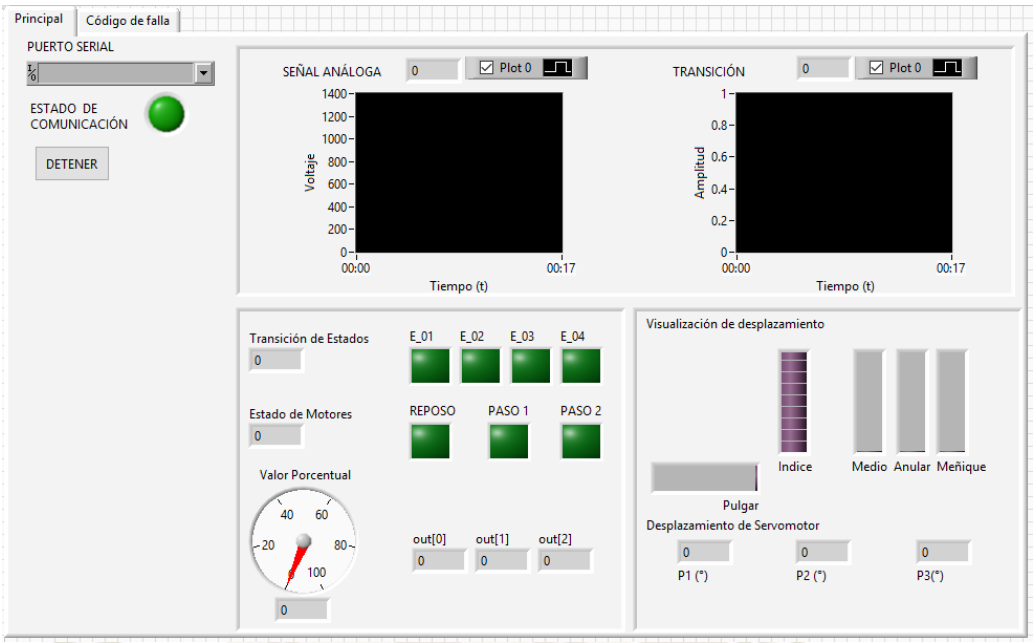


Figura 3.49: Interfaz gráfica para el usuario

Fuente: Elaboración propia

Al finalizar el desarrollo del programa se requiere el desarrollo de un archivo ejecutable para que se ejecuten sin necesidad de abrir el software LABVIEW y proteger la programación del usuario.

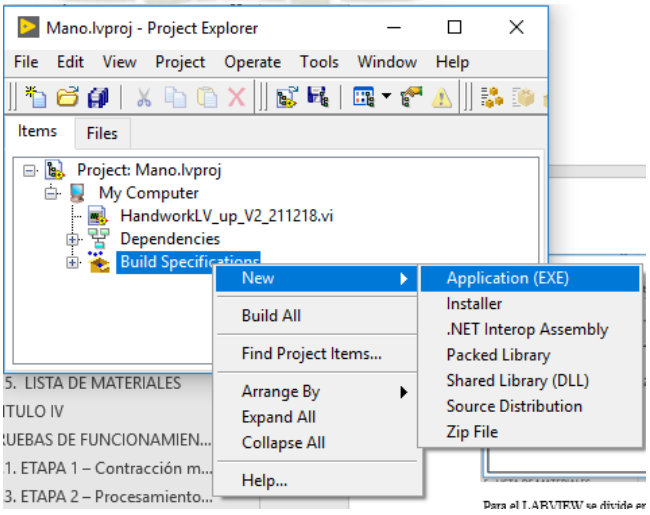


Figura 3.50: Creación de ejecutable

Fuente: Elaboración propia

3.4. HARDWARE

El hardware es parte física del proyecto, se va encargar de ejecutar y corroborar que el sistema funcione correctamente al interactuar con la programación.

La etapa está dividida en dos partes:

1. Electrónica
2. Mecánica

En la electrónica se idéntica el total de señales a usar, se realiza los cálculos respectivos y se diseñó la tarjeta electrónica que servirá de medio para la interactuación en el hardware mecánico y la programación.

En la mecánica la mano selecciona se implementará para poder realizar las pruebas del prototipo para realizar la impresión de las piezas con tecnología de impresión 3D.

3.4.1. ELECTRÓNICA

En esta etapa las consideraciones de diseño se basan en sistema de visualización principal, un sistema de sensado, sistema de potencia, controles opcionales y el sistema de monitoreo que son implementados en el prototipo a ser diseñado.

Para eso tenemos que tener claro cuáles son las señales de entrada y salida para realizar el diseño físico (Figura 3.51).

En la entrada que poseemos de forma física se va a dividir en dos módulos: primero es el módulo del sensado IR y en segundo lugar son los pulsadores, que servirán según sea configurado para su uso.

En la salida se divide en tres módulos: en primer lugar, son los servomotores que se cargan de desplazar cada dedo, en segundo lugar, es la comunicación serial que servirá para realizar el DEBUGGER y la transmisión de datos para la visualización de ser necesario y en último lugar se encuentra los leds que son la interacción visual del proyecto con el usuario final y programador.

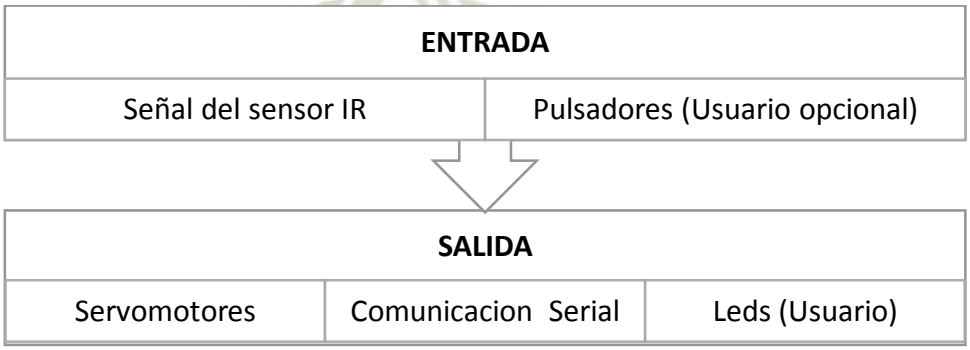


Figura 3.51: Secuencias de las señales

Fuente: Elaboración propia

3.4.1.1. ETAPA DE CONTROL Y MONITOREO

En esta etapa seleccionamos el microcontrolador a utilizar en base a las señales utilizas, el tipo, cantidad de señales, entre otros, que serán determinados en las tablas siguientes, y en base realizar el acondicionamiento de los sensores y actuadores. Finalmente, durante la etapa de monitoreo dispondremos de leds para que el usuario final pueda verificar que cada condición logré cumplirse y desarrollarse de forma correcta.

Primero identificamos los valores de entrada y salida del sistema principal en la tabla N°8.

Tabla N° 8: Tabla de identificación principal de señales

Componente	Representa	Rango	Señal	Tipo
Sensor IR	Lectura	Configurable	Análoga	Entrada
Servomotor	Pulgar	0-90	PWM	Salida
Servomotor	Índice	0-90	PWM	Salida
Servomotor	Otros	20-90	PWM	Salida

Fuente: Elaboración propia

También unas señales que se van a configurar para el usuario, la cuales servirán para ver el comportamiento de forma digital, es decir de encendido y apagado como también de dar condición de entradas que el programador desee configurar para facilitar la practica al usuario (Tabla N°9).

Tabla N° 9: Tabla de identificación de señales visibles e interacción

Componente	Representa	Rango	Señal	Tipo
Led	Servo 1	Booleana	Digital	Salida
Led	Servo 2	Booleana	Digital	Salida
Led	Servo 3	Booleana	Digital	Salida
Led	Estado 1	Booleana	Digital	Salida
Led	Estado 2	Booleana	Digital	Salida
Led	Estado 3	Booleana	Digital	Salida
Led	Estado 4	Booleana	Digital	Salida
Pulsador	Opcional	Booleana	Digital	Entrada
Pulsador	Opcional	Booleana	Digital	Entrada
Pulsador	Opcional	Booleana	Digital	Entrada
Pulsador	Opcional	Booleana	Digital	Entrada

Fuente: Elaboración propia

3.4.1.1.1. Selección del microcontrolador

Antes que todo para selección del microntrolador, tome como base los módulos electrónicos que interactúen con IDE ARDUINO. Esta presenta una serie de ventajas ya conocidas, como por ejemplo reducción de tiempo en el desarrollo de proyecto gracias a una serie de librerías y funciones predesarrollo, también este es de código abierto que da la posibilidad realizar copias y variantes.

		Apollo 11	IBM PC	Arduino Uno	chipKIT Uno32	Wiring S	Maple	LaunchPad MSP430G2553	LaunchPad Tiva C	Connected LaunchPad	Teensy 3.0
Year		1969	1981	Today	Today	Today	Today	Today			Today
Processor			Intel 8088	ATmega328	PIC32 MX320F128	ATmega644	ARM Cortex M3 STM32F103RB	MSP430G2553	ARM Cortex M4 LM4F120 / TM4C123	ARM Cortex M4 TM4C129	ARM Cortex M4 MK20DX128
Speed	**MHz**	1,024	4,77	16	80	16	72	16	80	120	48
Flash Memory	**KB**			32	128	64	128	16	256	1024	128
SRAM	**KB**	2	16	2	16	4	16	0,5	32	256	
Mass Storage	**KB**	32	360, 5 ¼" floppy	optional (1)	optional (1)	optional (1)	optional (1)	optional (1)	optional (1b)	optional (1b)	optional (1b)
Display		13 lights + 3x5 digits + 3x2 digits	320x240 pixels x 16 colours	optional (2)	optional (2)	optional (2)	optional (2)	optional (2)	optional (2b)	optional (2b)	optional (2b)
Input Device		19-key keypad	83-key keyboard	optional (3)	optional (3)	optional (3)	optional (3)	optional (3)	optional (3b)	optional (3b)	optional (3b)
Initial Price	**USD**	n/a	3 000								
Equivalent Price	**USD**	n/a	7 700	25	27	27	45	10	13	20	19

Figura 3.52: Comparación de características técnicas en diferentes plataformas

Fuente: https://embeddedcomputing.weebly.com, Selección de plataforma

Con lo desarrollado en las tablas N°8 y N°9. Obtendremos que se manejaran un total de 11 señales digitales, de las cuales 7 son de salida y 4 son de entrada, 1 señal análoga y 3 señales PWM, dando como requisito mínimo esta cantidad de señales.

Con una imagen comparativa e interpolando con las diferentes señales de entrada y salida, obtendremos por relación de características bastantes altas y una buena relación de precio al obtenerse al valor de 13 dólares americanos por lo cual se selecciona la plataforma de TIVA C.

También que el máximo consumo eléctrico del microcontrolador con todos sus puertos activos incluyendo la comunicación serial según su hoja de datos es de 250 mA, este dado sirve para la el cálculo de consumo de alimentación en el siguiente apartado.

Ahora se procede a la selección que pines de los puertos manejaran cada señal en este microcontrolador y plataforma.

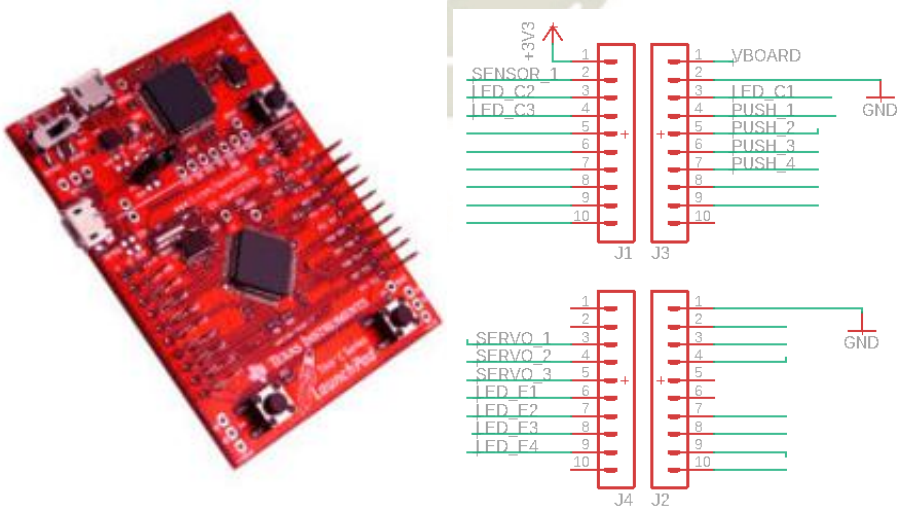


Figura 3.53: Pines e imagen del TIVA C

Fuente: Página web Texas Instrument, EK-TM4C123GXL

Se seleccionas los pines del microcontrolador con las respectivas señales para el sistema de control principal para enviar los comandos (Tabla N°10).

Tabla N° 10: Selección de pines para sensor y actuadores principales

Componente	Pin	Etiqueta
Sensor IR	PB_5	SENSOR_1
Servomotor	PD_6	SERVO_1
Servomotor	PC_7	SERVO_2
Servomotor	PC_6	SERVO_3

Fuente: Elaboración propia

Ahora seleccionamos la señal de control; es decir; los pines con los cuales nos van a mandar y recibir las señales del microcontrolador según el tipo y la señal (Tabla N°11).

Tabla N° 11: Tabla de selección de pines para las señales de iteración

Componente	Pin	Etiqueta
Led	PD_0	LED_C1
Led	PB_0	LED_C2
Led	PB_1	LED_C3
Led	PC_5	LED_E1
Led	PC_4	LED_E2
Led	PB_3	LED_E3
Led	PF_3	LED_E4
Pulsador	PD_1	PUSH_1
Pulsador	PD_2	PUSH_2
Pulsador	PD_3	PUSH_3
Pulsador	PE_1	PUSH_4

Fuente: Elaboración propia

3.4.1.1.2. SEÑAL DE SENSADO (IR)

En el circuito de la figura 3.54, a la entrada del microcontrolador con lectura análoga se acopla ópticamente al emisor del fototransistor del

sensor reflectivo infrarrojo QRE1113[8]. Este circuito consta de un LED y un fototransistor [14].

Se seleccionó este sensor debido a su diseño compacto y actualmente maneja un amplio uso en robots velocistas además que presenta un mejor desempeño que los famosos CNY70 o TCRT5000, teniendo como beneficio fácil integración con diferentes microcontroladores.[9]

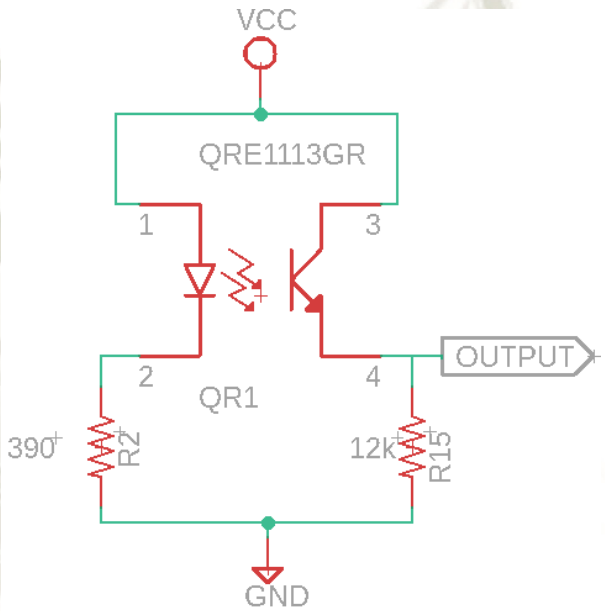


Figura 3.54: Circuito electrónico del sensor IR

Fuente: Elaboración propia

Datos:

$$VCC = 3.3VDC$$

$$R2 = 390\Omega \qquad R15 = 12k\Omega$$

Aplicando la ley de ohm para obtención de la corriente máxima para el consumo del circuito del Sensor IR.

[8] Hoja de datos del QR_QRE1113
[9] https://naylampmechatronics.com, sensor infrarrojo SMD qre1113

Obteniendo I_1:

$$I_1 = \frac{VCC}{R2}$$

$$I_1 = \frac{3.3VDC}{390\Omega}$$

$$I_1 = 8.46mA$$

Obteniendo I_2:

$$I_2 = \frac{VCC}{R15}$$

$$I_2 = \frac{3.3VDC}{12000\Omega}$$

$$I_2 = 0.275mA$$

Ahora calculamos la corriente consumida por el circuito del Sensor IR.

$$I_{Max\_Total} = I_1 + I_2$$

$$I_{Max\_Total} = 8.735\ mA$$

3.4.1.1.3. SERVOMOTORES

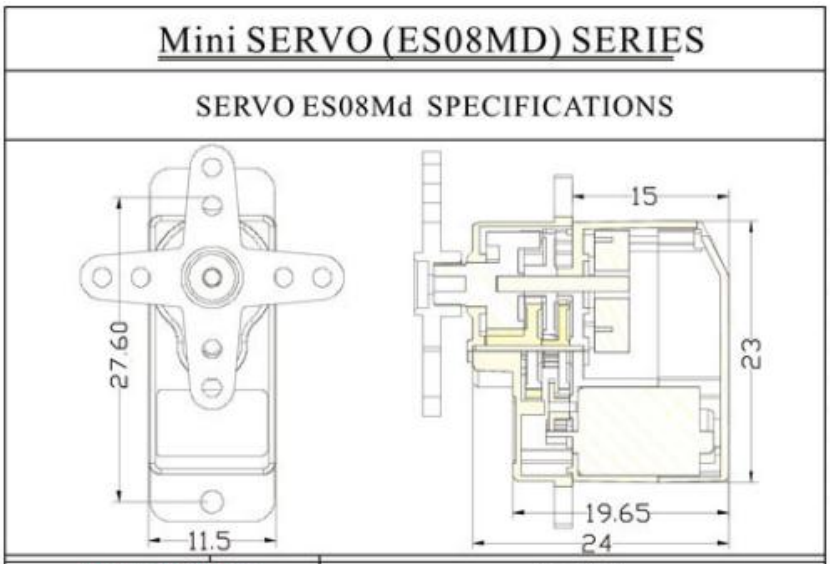
El sistema de control para los se instaló unos filtros con condensadores de 100nF cerca de la alimentación para los picos de corrientes que se genera en los motores DC. Siendo este un sistema de protección.

El control este dado al manipular de ancho de pulso que es entregado por el microntrolador con un rango de voltaje de 0V a 3.3V, variando un ciclo de trabajo.

Siendo un total de cuatro servomotores ES08MD y uno MG946R.

Estas se seleccionaron por recomendación del espacio establecido en la mano robótica HACKBERRY.

a. Servomotor ES08MD



Operating Voltage:		4.8~6.0V
STD Direction:		Counter Clockwise / Pulse Traveling 1500 to 1900usec
Stall Torque:	4.8V	1.6 Kgf.cm(21.0 oz.in)
Operating Speed:	4.8V	0.12 Sec/60° at no load
Stall Torque:	6.0V	2.0 Kgf.cm(28.0 oz.in)
Operating Speed:	6.0V	0.10 Sec/60° at no load
Size:		23X11.5X24 mm
Weight:		12 g
Plug Available		FUT ;JR
Other		Digital; Metal

Figura 3.55: Hoja Características ES08MD

Fuente: https://www.seeedstudio.com, EMax-12g-ES08MD

Datos:

Voltaje: 4.8v.

Velocidad: 0.12 sec/60º.

Torque: 1.6 Kgf.cm a 4.8v.

Primero realizamos la conversión de unidades al torque de $Kgf.cm$ a Nm:

$$T = 1.6\ Kgf.cm * \frac{9.81N}{1KgF} * \frac{1m}{100cm}$$

$$T = 0.15696\ N.m$$

Luego realizamos la conversión de unidad a la velocidad angular (w):

$$w = \frac{rad}{seg}$$

$$w = \frac{60 * \frac{\pi}{180}}{0.12}$$

$$w = 8.75\ \frac{rad}{seg}$$

Una vez hemos obtenido la velocidad angular en las unidades requeridas, calcularemos la potencia que demanda el servomotor aplicando la fórmula de:

$$POTENCIAS\ (P) = TORQUE(T) * VELOCIDADANGULAR\ (w)$$

$$P = T * w$$

$$P = 0.15696 * 8.75$$

$$P = 1.37\ W$$

Ahora calculamos la corriente que consume cada servomotor ES08MD

$$POTENCIA(P) = VOLTAJE(V) * CORRIENTE(I)$$

$$P = V * I$$

Despejando la corriente (I) y reemplazando las variables obtenemos el valor deseado:

$$I = \frac{P}{V}$$

$$I = \frac{1.37\ W}{4.8\ V}$$

$$I = 285.4\ mA$$

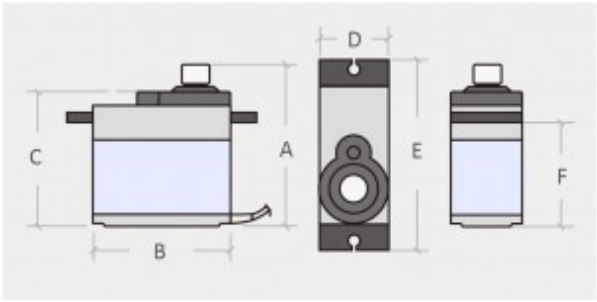
Ahora calculamos la corriente máxima consumida por los 2 servomotores.

$$I_{Total} = I * N_{Servomotores\_\text{ES08MD}}$$

$$I_{Total} = 285.4 * 2$$

$$I_{Total} = 570.8\ mA$$

b. Servomotor MG946R



PRODUCT CONFIGURE TABLE

Weight(g)	55
Torque(kg)(4.8v)	10.5
Speed(sec/60deg)	0.2
A(mm)	42.7
B(mm)	40.9
C(mm)	37
D(mm)	20
E(mm)	54
F(mm)	26.8

Figura 3.56: Hoja Características MG946R

Fuente: http://www.towerpro.com.tw, MG946R

Datos:

Voltaje: 4.8v.

Velocidad: 0.2 sec/60º.

Torque: 10.5 Kgf.cm a 4.8v.

Primero realizamos la conversión de unidades al torque a Nm:

$$T = 10.5\ Kgf.cm * \frac{9.81N}{1KgF} * \frac{1m}{100cm}$$

$$T = 1.03\ N.m$$

Luego realizamos la conversión de unidad a la velocidad angular (w):

$$w = \frac{rad}{seg}$$

$$w = \frac{60 * \frac{\pi}{180}}{0.2}$$

$$w = 5.235\ \frac{rad}{seg}$$

Una vez hemos obtenido la velocidad angular en las unidades requeridas, calcularemos la potencia que demanda el servomotor aplicando la fórmula de:

$$POTENCIAS\ (P) = TORQUE(T) * VELOCIDADANGULAR\ (w)$$

$$P = T * w$$

$$P = 1.03 * 5.235$$

$$P = 5.392\ W$$

Ahora calculamos la corriente que consume cada servomotor MG946R

$$POTENCIA(P)\ = VOLTAJE(V) * CORRIENTE(I)$$

$$P\ = V * I$$

Despejando la corriente (I) y reemplazando las variables obtenemos el valor deseado:

$$I = \frac{P}{V}$$

$$I = \frac{5.392\ W}{4.8\ V}$$

$$I =\ 1.12\ A$$

3.4.1.1.4. PULSADORES

En los pines de conexionado 1 y 2 de cada pulsador se conectan al microcontrolador y a una fuente alimentación, para que el microcontrolador tenga un nivel de referencia inicial de 3.3VDC. entonces se configura (Figura 3.57) de forma de resistencia a pulsador, logrando un comportamiento que, al accionarse el pulsador, la señal de 3.3VDC pase a través de la resistencia y así obtener una señal de referencia de potencial "0" (GND) en los pines de lectura digital del microcontrolador.

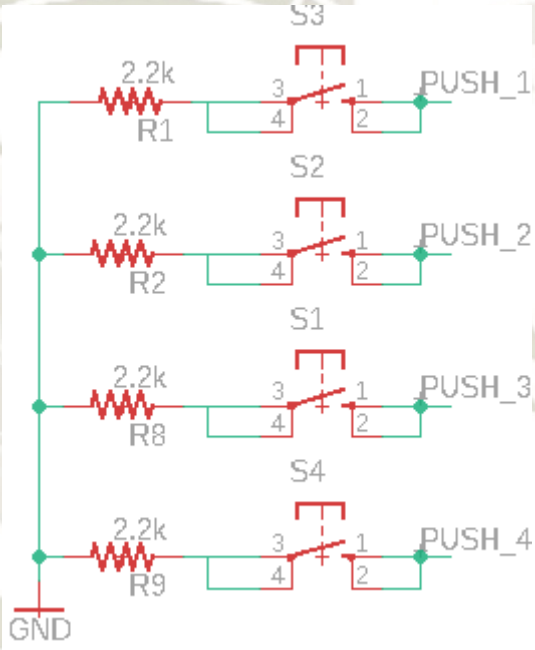


Figura 3.57: Esquema electrónico de pulsadores

Fuente: Elaboración propia

Datos:

$$V_{max} = 3.3VDC$$

$$R = 2.2k\Omega \qquad N_{Puldadores} = 4$$

Aplicando la ley de ohm para obtención de la corriente máxima para el consumo de la línea de pulsador-resistencia.

$$I = \frac{V}{R}$$

$$I = \frac{3.3VDC}{2200\Omega}$$

$$I = 1.65mA$$

Ahora calculamos la corriente consumida por los 4 pulsadores.

$$I_{Total} = I * N_{Pulsadores}$$

$$I_{Total} = 1.65 * 4$$

$$I_{Total} = 6.6mA$$

3.4.1.1.5. MONITOREO

Tiene dos sistemas de monitoreo, uno opcional siendo la comunicación serial hacia el software LABVIEW y el que se visualiza con leds con el cual vemos la transición de estados de cada motor y transición de lógica de funcionamiento.

La fórmula que se aplica para este cálculo es la ley de ohm con la cual obtener la corriente de consumo de cada LED.

El valor resistivo que selecciono es de 2.2kΩ.

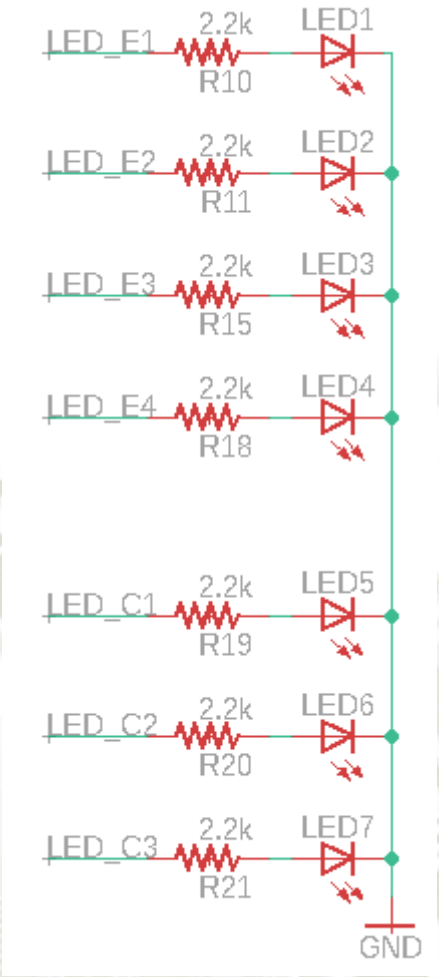


Figura 3.58: Circuito de Led para estados

Fuente: Elaboración propia

Datos:

$$V_{max} = 3.3VDC$$

$$R = 2.2k\Omega$$

$$N_{leds} = 7$$

Aplicando la ley de ohm para obtención de la corriente para el consumo de cada led.

$$I = \frac{V}{R}$$

$$I = \frac{3.3VDC}{2200\Omega}$$

$$I = 1.65mA$$

Ahora calculamos la corriente consumida por los 7 LEDS.

$$I_{Total} = I * N_{leds}$$

$$I_{Total} = 1.65 * 7$$

$$I_{Total} = 11.55mA$$

3.4.1.2. ETAPA DE POTENCIA

Para la etapa de potencia de nuestro sistema tenemos los servomotores que se van a controlar y al sistema de alimentación como todo el consumo del sistema en general (Figura 3.59).

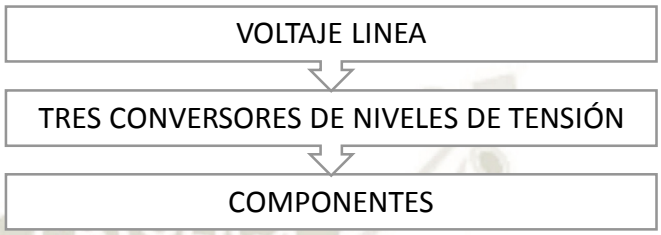


Figura 3.59: Etapa de potencia

Fuente: Elaboración propia

3.4.1.2.1. ALIMENTACIÓN

Para la etapa de potencia de nuestro sistema se requiere tres convertidores de nivel de tensión (Figura 3.60), iniciamos con un convertidor AC-DC que realizar el cambio de 220 VAC a 19 VDC, luego pasa a otro convertidor DC-DC que reduce el voltaje a 5 VDC que alimenta a los tres servomotores y al TIVA C, que trabajo con el sistema ARDUINO, también tiene un convertidor interno adicional que vuelve a reducir al nivel de 3.3 VDC que alimenta al microcontrolador, a los sistemas físicos de monitoreo y finalmente al sensor IR.

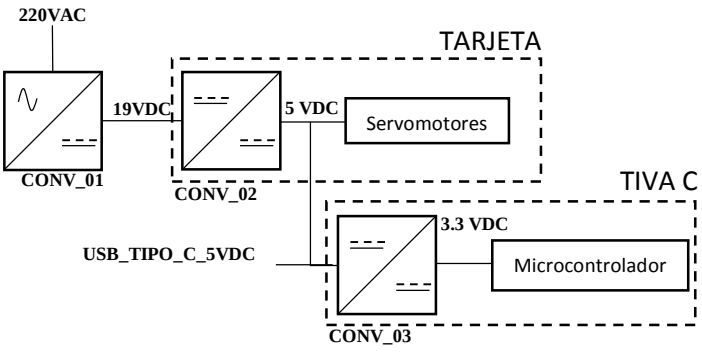


Figura 3.60: Estados de los niveles de tensión

Fuente: Elaboración propia

En la salida del regulador (Figura 3.61) de 5 VDC presenta dos diodos, uno va a los servomotores y el otro a la tarjeta TIVA C, esto se usa para evitar la reversión de corriente de un sistema a otro, ya que el TIVA C tiene conexión USB-C el cual también tiene una alimenta al sistema con 5VDC, pero con limitador de corriente menor, por lo cual es necesario proteger ambos sistemas de alimentación.

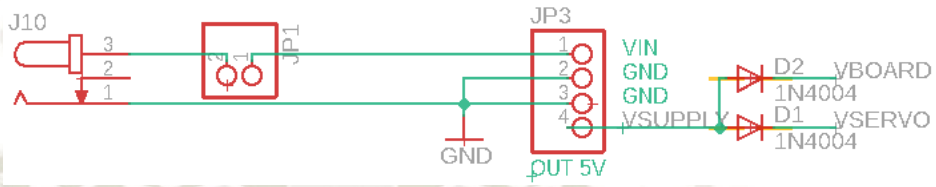


Figura 3.61: Alimentación con diodos de protección

Fuente: Elaboración propia

Para seleccionar esta fuente de alimentación se basó en la suma del consumo global, asumiendo que todo el sistema está funcionando en paralelo, esto incluye el máximo consumo del microcontrolador al activar todos sus puertos I/O y los de transmisión de datos análogos y digitales siendo un valor de 250ma (este dato obtenido de la hoja de datos del microcontrolador), además de adicionar un factor de seguridad (FS) de 1.2 al consumo.

Aplicamos la sumatoria de las corrientes obtenidas en la etapa de control y monitoreo 3.4.1.1.

$$I_{total} = I_{IR} + I_{Total_{ES08MD}} + I_{MG946R} + I_{Total_{pulsadores}} + I_{Total_{leds}} + I_{Microcontrolador}$$

$$I_{total} = 8.735\ \text{mA} + 570.8\ \text{mA} + 1.12\ \text{A} + 6.6\text{mA} + 11.55\text{mA} + 250\text{mA}$$

Obtenemos la corriente máxima consumida:

$$I_{total} = 1967.685 \text{ mA} \cong 1.97 \text{ A}$$

Finalmente aplicamos a la corriente máxima el factor de seguridad de 1.2 para obtener la corriente que mínima para el regulador a implementar.

$$\mathrm{I_{Alimentación}} = \mathrm{I_{total}} * FS$$

$$I_{Alimentación} = 1.97\ A * 1.2$$

$$I_{Alimentación} = 2.364\ A$$

Con lo cual validamos la selección del conversor de voltaje POLOLU D24V25F5 (Figura 3.62).

Para el conversor de 19VDC a 5VDC se utilizó el regulador de voltaje POLOLU 5VDC (Figura 3.62), que entrega un máximo de 2.5A.

Figura 3.62: Regulador de voltaje Pololu 5V, 2.5A D24V25F5

Fuente: Pololu Web

Los actuadores presentan un mayor consumo de corriente, ya que teóricamente la alimentación del microcontrolador es fija, como sabemos cada servomotor (Figura 3.63) tiene su sistema de control interno con el cual controla el motor DC por lo cual solo hay que alimentarlo y enviar la señal de control.

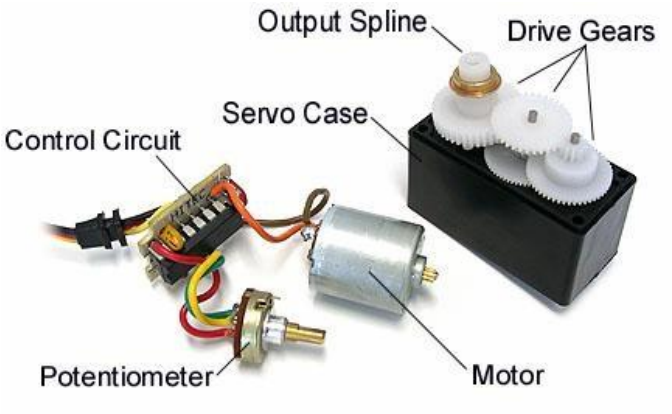


Figura 3.63: Partes del servo motor

Fuente: http://cytron.com.my

3.4.2. MECÁNICA

Para el sistema mecánico se realizó tres ensambles:

1. Mano
2. Encapsulado de la electrónica.
3. Encapsulado del IR

A todos los archivos 3D generados por los softwares de diseño requieren una conversión de formato para poder realizar la impresión siendo el formato STL también conocido como "objeto 3D", luego con la ayuda del programa REPETIER-HOST de nuestra impresora y la impresora 3D, nos da la facilidad de materializar el proyecto de la mano robótica.

Antes de poder realizar lo anterior se requiere seleccionar el hardware mecánico físico, el hardware mecánico seleccionado es el HACKBERRY desarrollado por la compañía EXIII (Figura 3.64) que tiene un beneficio muy importante debido a que está desarrollada en código abierto, esto quiere decir que este trabajo fue desarrollado por un grupo de ingenieros y se encuentra disponible en su página web, dando la facilidad de poder replicar el proyecto o modificarlo según sea conveniente para el que lo use.

Figura 3.64: Modelo HACKBERRY[10]

Fuente: http://exiii-hackberry.com

[10] http://exiii-hackberry.com/

A los archivos se puede acceder en la web de GITHUB (Figura 3.65) de la compañía, donde desarrollaron el proyecto completo del cual se imprimió la mano izquierda (Left).

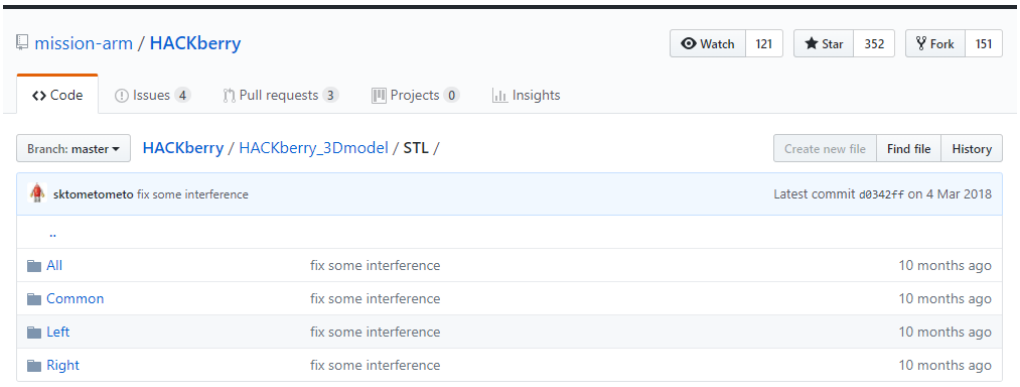


Figura 3.65: GITHUB del modelo mecánico

Fuente: https://github.com, HACKberry modelo 3D

El software que se usó para la materialización del proyecto es conocido como RepetierHost desarrollado por una compañía alemana, siendo esta usada por la impresora 3D PRUSA I3.

Figura 3.66: Programa Repetier Host

Fuente: Inicio del programa Repetier Host

Este software es bastante práctico e intuitivo, se conecta a la impresora 3D por el puerto serial, en este programa importamos el STL del diseño mecánico para luego

ser compilado y luego enviar los comandos a los motores paso a paso de la impresora 3D.

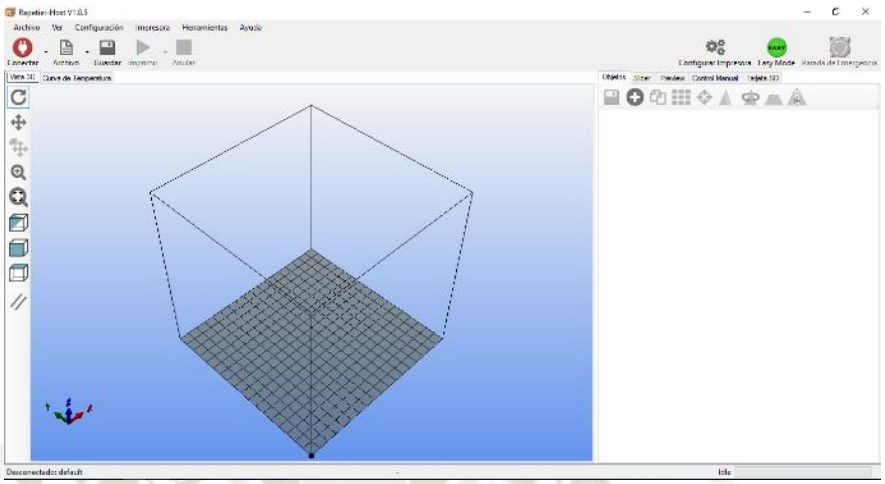


Figura 3.67: Interfaz del Repetier Host

Fuente: Programa Repetier Host

Se realizó la importación al programa REPETIERHOST de todos los archivos STL solo fines de visualización, ya que de forma real para esta impresora se imprimió de uno a uno, es decir, una pieza a la vez.

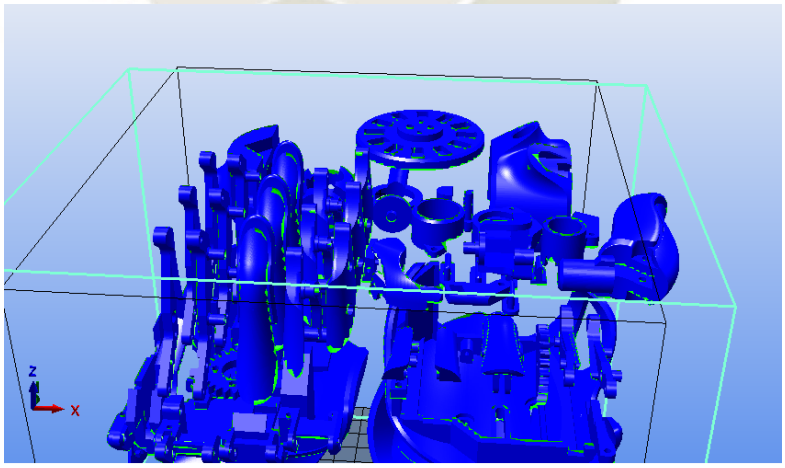


Figura 3.68: Diseño 3D de la mano robótica

Fuente: Importación del ensamble completo de la mano HACKBERRY

La impresora 3D que se usó se encuentra configurada para imprimir plástico del tipo PLA (Debido a que este material no produce gases nocivos en comparación

del ABS) además de poderlo imprimir a temperaturas no altas por lo cual se compró este material para desarrollar el proyecto.

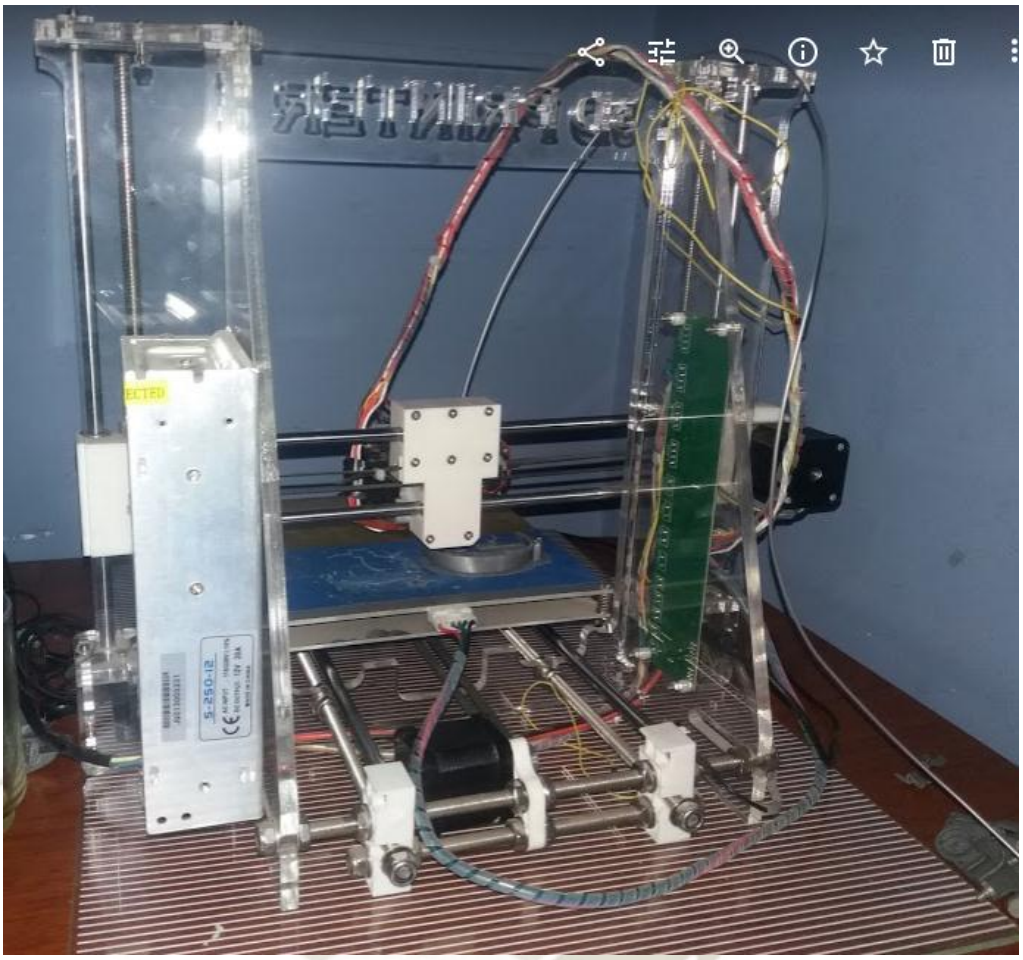


Figura 3.69: Impresión 3D en Prusa I3

Fuente: Fotografía de la impresora 3D Prusa I3

El archivo 3D de la mano HACKBERRY se importó y se ejecutó en el programa de SOLIDWORKS para poder mostrar el ensamble 3D de la mano antes de realizar la respectiva impresión, poder corroborar y comprobar el diseño de la mano, además de contrastar con el sistema real.

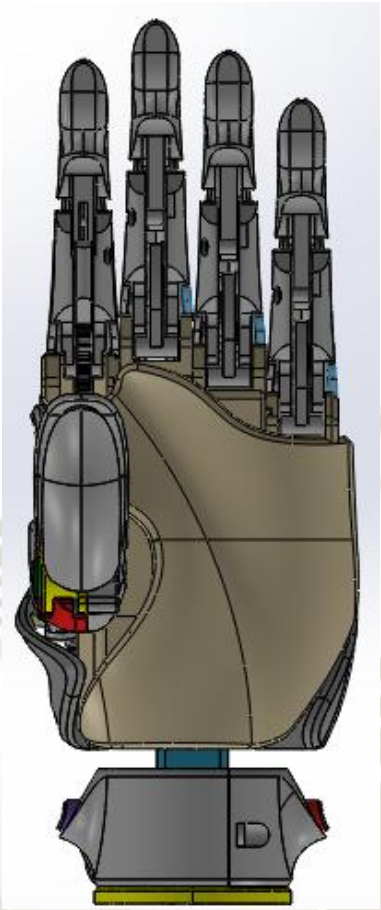


Figura 3.70: Diseño 3D de la mano robótica

Fuente: Modelo 3D importado en SOLIDWORKS

La mano 3D consta de 61 piezas (Tabla N°12):

Tabla N° 12: Piezas 3D de la mano

Componente	Cantidad
Encapsulado mano	15
Índice	7
Meñique, anular, medio	15
Socket	10
Pulgar	6
Muñeca	8

Fuente: Elaboración propia

Luego de realizar la impresión de la mano se procede a realizar el ensamble, las cuales se mostrarán con las tres diferentes vistas antes de realizar el ensamble al soporte: Lateral (Figura 3.71), Posterior (Figura 3.72) y vista frontal del ensamble (Figura 3.73).

Figura 3.71: Vista lateral del ensamble de la mano robótica

Fuente: Elaboración propia

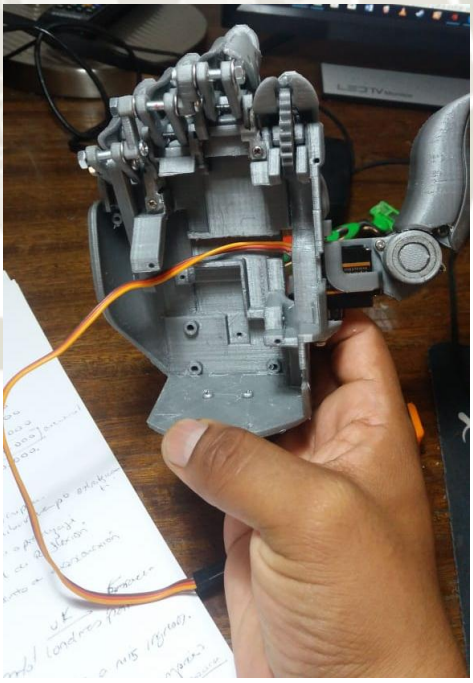


Figura 3.72: Vista Posterior interior de la mano robótica

Fuente: Elaboración propia

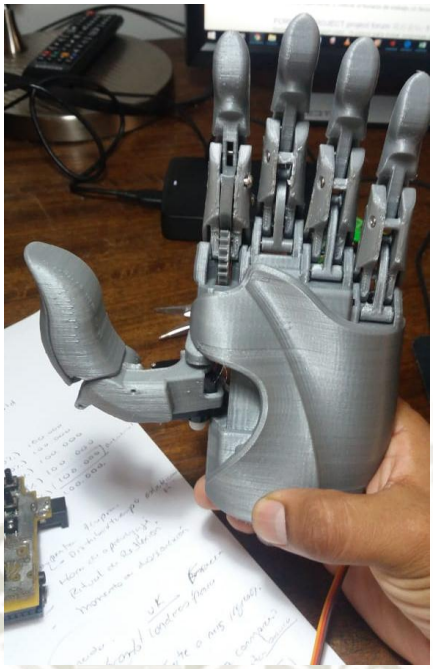


Figura 3.73: Vista frontal de la mano robótica

Fuente: Elaboración propia

Posteriormente se realiza el ensamble del soporte de la mano robótica Figura 3.74 y Figura 3.75

Figura 3.74: Piezas preensamble de soporte de la mano robótica

Fuente: Elaboración propia

Figura 3.75: Soporte de la mano robótica

Fuente: Elaboración propia

El módulo electrónico de la mano se atornilla a una base de melanina junto con la mano, donde se instaló dos puntos de anclaje (figura 3.76), este fue diseñado en base a la tarjeta electrónica en el programa de SOLIDWORKS, y al final para el desarrollo del proyecto comprobar que el modelo, diseño se asemeje y cumpla.

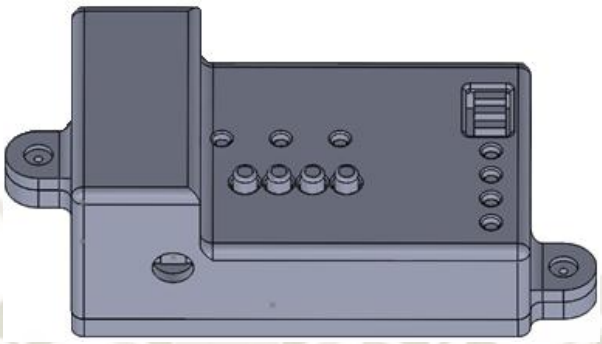


Figura 3.76: Módulo electrónico del sistema

Fuente: Modelo 3D SOLIDWORKS

El módulo 3D consta de seis piezas.

Tabla N° 13: Piezas 3D del módulo electrónico del sistema

Componente	Cantidad
Cubierta superior	1
Cubierta inferior	1
Botones	4

Fuente: Elaboración propia

Ya con la tarjeta electrónica y el modulo impreso se procede a realizar el ensamble (Figura 3.77)

Figura 3.77: Módulo electrónico ensamble e impresión 3D

Fuente: Elaboración propia

El encapsulado almacena la tarjeta de sensado IR y se ubica cerca al supinador del brazo, donde obtendremos la contracción muscular como también se conecta el cable de audio para el cual también se imprimió dos partes (Tabla N°14).

Figura 3.78: Encapsulado del IR

Fuente: Modelo 3D SOLIDWORKS

El encapsulado de IR consta de dos piezas.

Tabla N° 14: Piezas 3D del encapsulado IR

Componente	Cantidad
Cubierta superior	1
Cubierta inferior	1

Fuente: Elaboración propia

Se procede a realizar la impresión y posteriormente el ensamble respectivo con la tarjeta electrónica (Figura 3.79)

Figura 3.79: Ensamble de encapsulado IR

Fuente: Elaboración propia

Se instala el módulo electrónico y la mano robótica en la base de melamina para la demostración, además deben estar perfectamente anclado para realizar las pruebas de funcionamiento.

Figura 3.80: Módulo completo del proyecto ensamblado

Fuente: Elaboración propia

3.5. LISTA DE MATERIALES

Esta lista no incluye costos de envíos y no se considera compra a por mayor cantidad.

Tabla N° 15: Tabla de listas de materiales.

NOMBRE DE MATERIALES	DESCRIPCIÓN	CANTIDAD	COSTO (PEN)
Rodamiento	MF74ZZ 4*7*2.5mm	1	4
Pulsador táctil	Micro switch temporal 6*6*11mm	4	2
Servomotor	ES08MD II	2	46
Servomotor	MG946R	1	36
Regulador de Voltaje	Pololu 5V, 2.5A Step-Down Voltage Regulator D24V25F5	1	36
Rollo 3D PLA	Plateado 1.75mm 1kg	2	35
Cable USB	Tipo C a tipo A	1	5
Cable de audio	3.5mm Macho-Macho 1m	1	5
LED	Led Azul	3	0.30
LED	Led Verde	4	0.40
QRE1113	SMD	1	3
TM4C123G LaunchPad	TM4C123G	1	43
Conector de alimentación	2.1 mm	1	1
Tuerca	M3	50	6
Perno	M3, 25mm	50	12
Resistencia	2k de 1/4W	7	0.3
Resistencia	10k de 1/4W	4	0.2
Resistencia	390 de 1/4W	1	0.1
Resistencia	12k de 1/4W	1	0.1
Condensador	100nf	3	0.5
Espadines	Hembra	5	5
Espadines	Macho	2	2
Diodo	RL205 2A	2	1
		Total	**243.9 PEN**

Fuente: Elaboración propia

CAPITULO IV
PRUEBAS DE FUNCIONAMIENTO

CAPITULO IV

PRUEBAS DE FUNCIONAMIENTO

Se realiza las pruebas de funcionamiento en la secuencia de cada etapa.

Para mantener un orden modular al proyecto se ubicarán las pruebas de forma secuencial en base a las etapas que presenta en el capítulo de desarrollo de ingeniería.

De forma física presenta tres ensambles mecánicos:

- Mano robótica: Luego de imprimir las piezas 3D en PLA, realizamos el ensamble del mismo.
- Modulo electrónico: También se imprime en la impresora 3D, las piezas del ensamble mecánico y la tarjeta electrónica fabricada para proseguir con el ensamblaje final.
- Módulo Sensor infrarojo: este módulo también consta de parte 3D y tarjeta electrónica, las cuales al finalizar de fabricarlos realizamos el ensamble final.

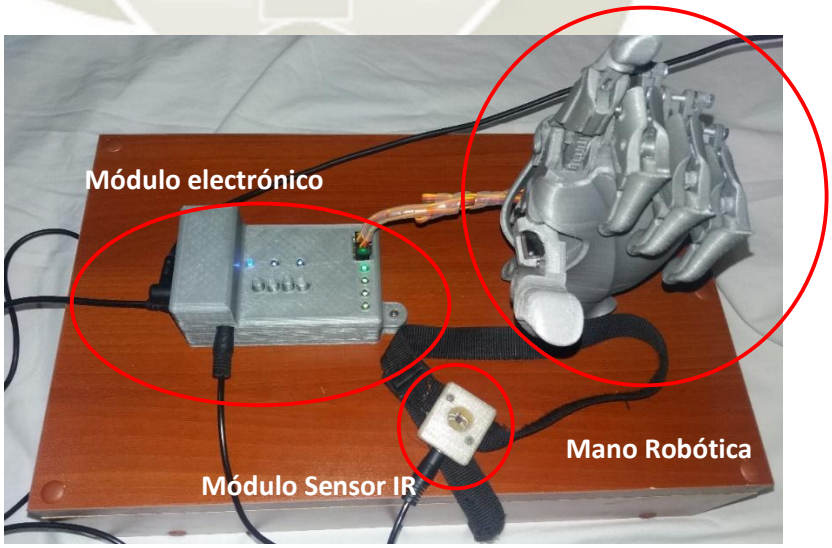


Figura 4.1: Módulo por OMG de una mano robótica con tecnología de impresión 3D

Fuente: Elaboración propia

4.1. ETAPA 1 – Contracción muscular

En la primera etapa verificamos que se desarrolle una correcta lectura análoga del sistema y también se realizó la verificación con el sistema de monitoreo corrobore que este desarrolle de forma correcta y entregue las señales en tiempo real.

Primero instalamos la banda al módulo del sensor IR (Figura 4.2), el cual se va a ubicar cerca al musculo pronador del antebrazo.

Este módulo hay que ubicarlo de forma adecuada para que el sensor infrarojo envíe la señal de la lectura correcta al microcontrolador luego de esto, manejamos dos lógicas básicas en el musculo, una que es mantener el musculo relajado con la cual envía un valor análogo mínimo (Figura 4.3).

Al contraer el musculo se bloquea parte de la luz IR que viaja hacia el musculo pronador del antebrazo y da un mayor retorno al fototransistor, elevando el valor analógico que ingresa al microcontrolador (Figura 4.4).

Figura 4.2: Banda con sensor IR.

Fuente: Elaboración propia

Figura 4.3: Músculo relajado

Fuente: Elaboración propia

Figura 4.4: Músculo contraído

Fuente: Elaboración propia

Al realizar la acción de mantener y soltar, manteniendo una condición inicial como el reposo del musculo y luego realizar la acción de contra el músculo, se puede obtener gráficamente la variación de voltaje en forma ascendiente (Figura 4.5).

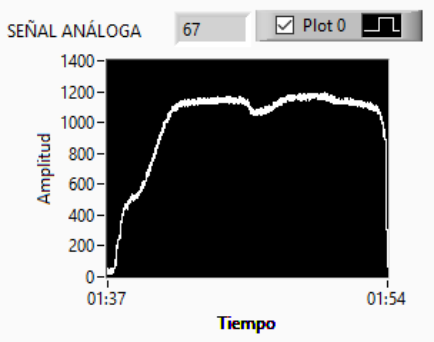


Figura 4.5: Respuesta análoga adquirida por el LABVIEW

Fuente: Elaboración propia

Al poder ya leer este valor análogo se realizar las pruebas de funcionamiento de lo que se desarrolló en la parte de programación en el 3.3.1, que fue la selección del valor adecuado para que el umbral realice el cambio a valores digitales (Figura 4.6).

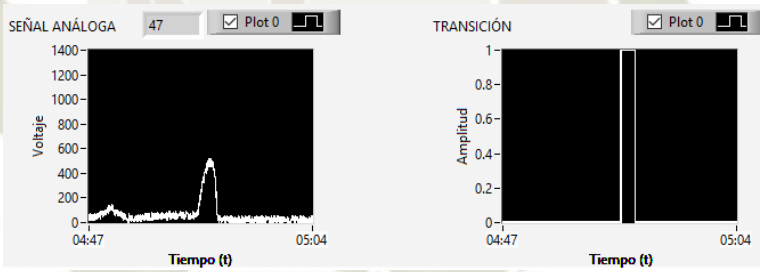


Figura 4.6: Acondicionado de los valores analógicos a digital

Fuente: Elaboración propia

4.2. ETAPA 2 – Procesamiento de Señal

En esta etapa de las pruebas de funcionamiento es corroborar lo que se desarrolló en el capítulo 3, siendo el tema principal como el desarrollo de la ingeniería y que cumpla con su funcionamiento planteado.

En esta etapa tiene como base detección de los "*triggers"* o pulsos:

1. CC: Co-contracción
2. DI: Doble impulso
3. TI: Triple impulso.
4. HO: Mantener-abrir.
5. DHO: Doble Mantener-abrir

Con los tres primeros se crean las señales para la lógica principal de funcionamiento y las dos últimas nos sirve para los sub-transiciones de desplazamiento como se explica en la sección 3.2.3. y la sección 3.3.3.

a. Señales para Lógica principal:

Con la ultimación del programa de LABVIEW logramos obtener las diversas graficas en las cuales se realiza del correcto comportamiento de estas señales.

En la figura 4.7 se corrobora la conversión a valor digital de una co-contracción hecha por el musculo y este siendo detectado durante el periodo de tiempo predeterminado

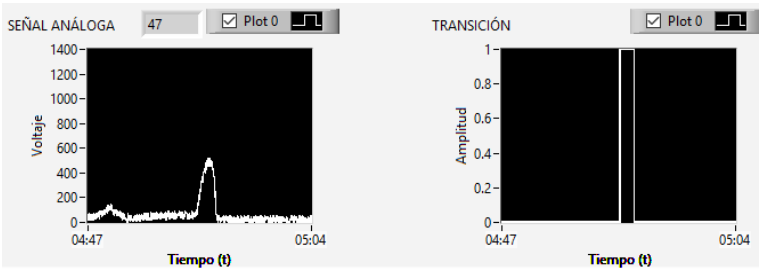


Figura 4.7: Acondicionado de los valores analógicos a digital de la co-contracción

Fuente: Elaboración propia

En la figura 4.8 se corrobora la conversión a valor digital de un doble impulso (DI) hecha por el musculo y este siendo detectado durante el periodo de tiempo predeterminado para cumplir la lógica.

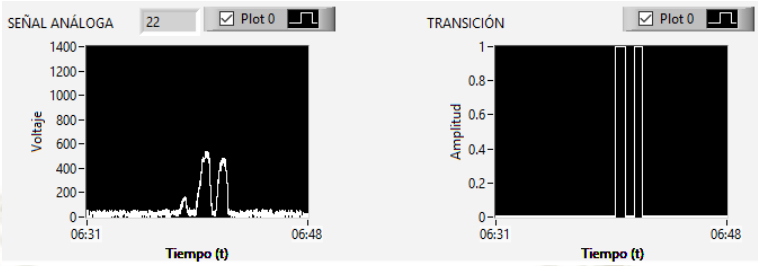


Figura 4.8: Acondicionado de los valores analógicos a digital del doble impulso (DI)

Fuente: Elaboración propia

En la figura 4.9 se corrobora la conversión a valor digital del triple impulso (TI) hecha por el músculo y este siendo detectado durante el periodo de tiempo predeterminado para cumplir la lógica.

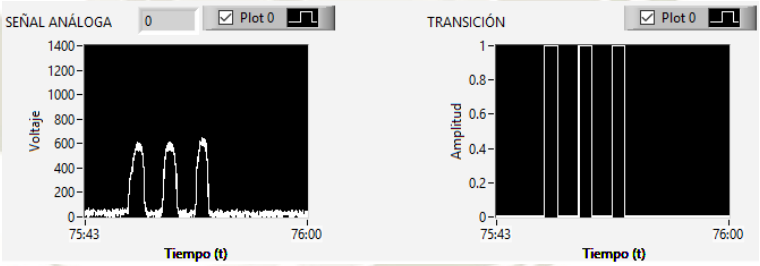


Figura 4.9: Acondicionado de los valores analógicos a digital del triple impulso (TI)

Fuente: Elaboración propia

Señales para Lógica interna:

Para poder realizar las 8 condiciones y no sea tan complicado para el usuario acceder a ellas se realizó antes mencionado cuatro estados y en cada uno presenta dos sub-estados o transiciones internas dándonos un total de 8 transiciones si considerar el estado por defecto. Las dos seleccionas para este caso son el HO y DHO.

En la figura 4.10 se corrobora la conversión del valor mantenido (HO) hecha por el musculo y este siendo detectado durante el periodo de tiempo predeterminado para cumplir la lógica.

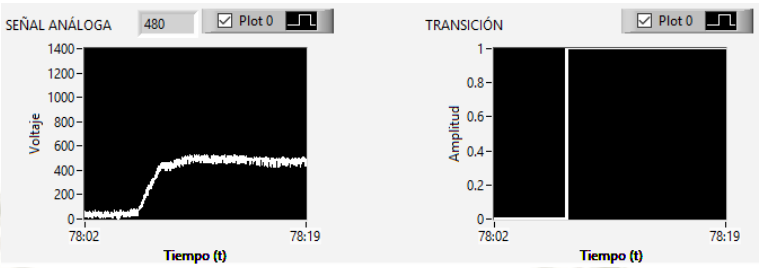


Figura 4.10: Acondicionado de los valores analógicos a digital del mantener y soltar (HO)

Fuente: Elaboración propia

En la figura 4.11 se corrobora la conversión del valor de doble mantener-soltar (DHO) hecha por el musculo y este siendo detectado durante el periodo de tiempo predeterminado para cumplir la lógica.

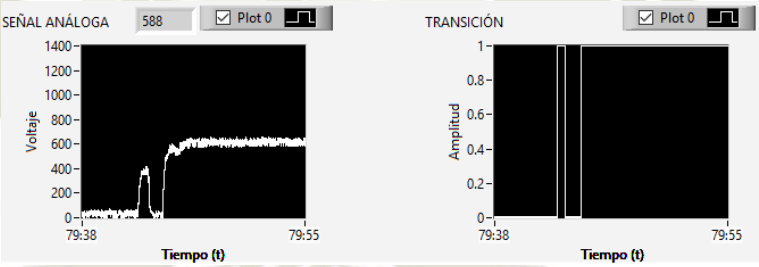


Figura 4.11: Acondicionado de los valores analógicos a digital del Doble mantener y soltar (DHO)

Fuente: Elaboración propia

Se utilizó el sistema LABVIEW para poder visualizar los parámetros con los respectivos comportamientos y corroborarlo además del módulo electrónico que se podrá visualizar de practica sin necesidad de acceder a un computador o al programa LABVIEW y realizar lo antes mencionado (Figura 4.12).

Figura 4.12: Acondicionado de los valores analógicos a digital de la co-contracción

Fuente: Elaboración propia

Los leds verdes son usados visualizar la transición de estados de la lógica principal y los leds azules para la transición de estados internos de forma lineal.

4.3. ETAPA 3 – Movimiento de los dedos

Para la realización de las pruebas de desplazamiento de los dedos con la lógica a implementada en el microcontrolador podemos verificar ya un adecuado funcionamiento.

Teniendo entendido que manejados cuatro estados y cada uno tiene tres sub-estados o Sub-transición de las cuales una es de condición de reposo, el comportamiento de cada uno se puede obtener de forma gráfica según la Figura 4.13.

Se realizó el control de las transiciones de estado cumpliendo adecuadamente la lógica de funcionamiento programada y adecuado desplazamiento de los motores, siendo verificado todos los estados.

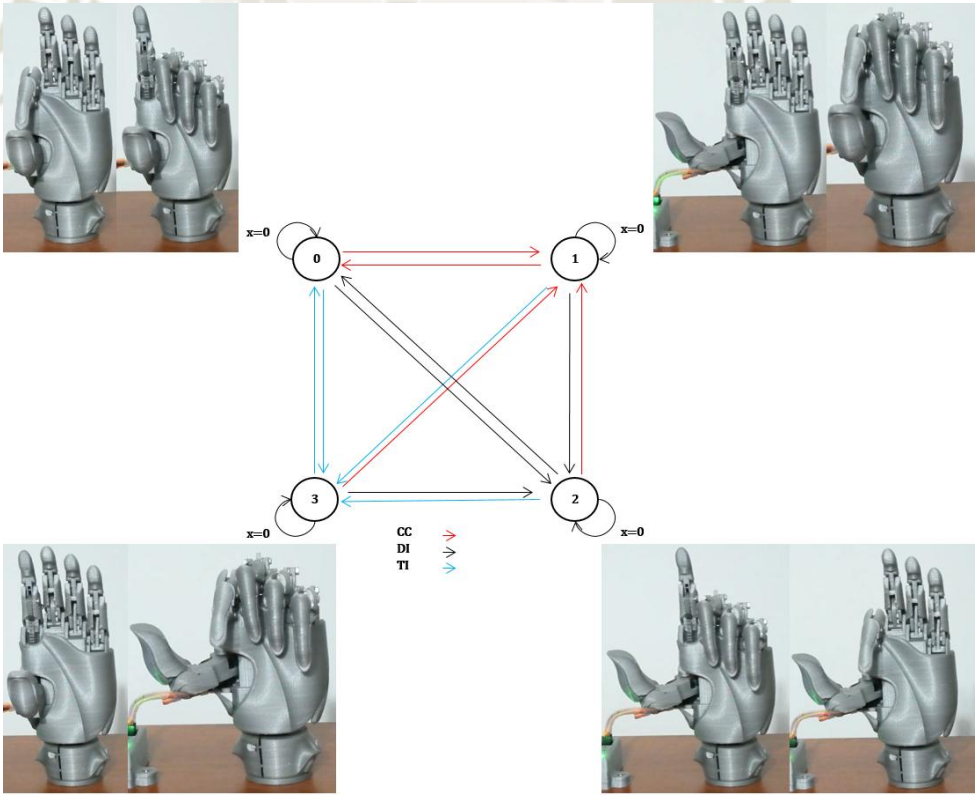


Figura 4.13: Transición de estados con desplazamiento final de los motores

Fuente: Elaboración propia

a. ESTADO 0

Para cumplir este estado verificamos que el **primer** led verde se encienda y se encuentre en el **primer** led azul, que es la condición de reposo de los servomotores.

Figura 4.14: Modulo electrónico en el primer estado principal

Fuente: Elaboración propia

Lógica interna: En la lógica interna, se realiza en la transición de dos estados para que se cumpla el primer estado es realizar la acción de HO y visualmente se iluminara el segundo led azul (Figura 4.15)

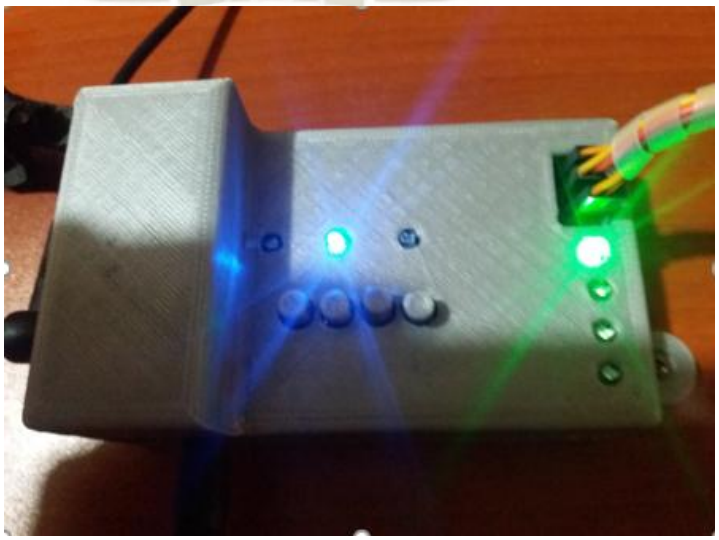


Figura 4.15: Transición interna del primer sub-estado

Fuente: Elaboración propia

En la figura 4.16. se corrobora el desplazamiento de los actuadores para el primer estado interno.

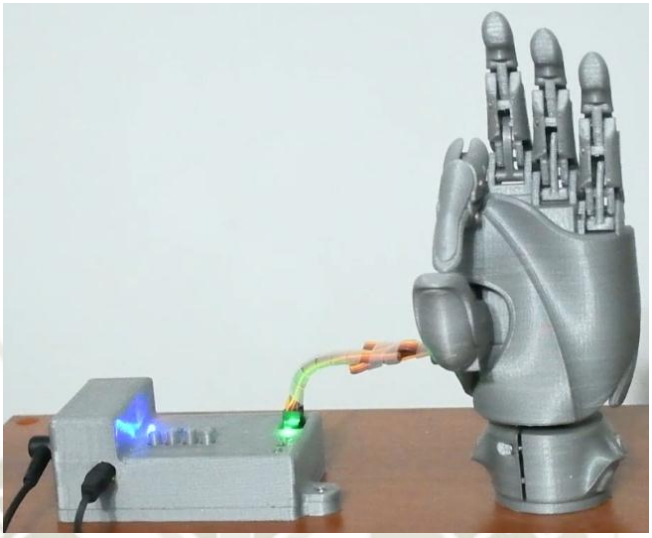


Figura 4.16: Desplazamiento de la mano del primer estado y primer estado interno

Fuente: Elaboración propia

Para que se cumpla el segundo estado es realizar la acción de DHO y visualmente se iluminara el tercer led azul (Figura 4.17)

Figura 4.17: Transición interna del segundo sub-estado

Fuente: Elaboración propia

En la figura 4.18. se corrobora el desplazamiento de los actuadores para el segundo estado interno.

Figura 4.18: Desplazamiento de la mano del primer estado y segundo estado interno

Fuente: Elaboración propia

b. ESTADO 1

Para cumplir este estado verificamos que el **segundo** led verde se encienda y se encuentre en el **primer** led azul también encendido, que es la condición de reposo de los servomotores.

Figura 4.19: Modulo electrónico en el segundo estado principal

Fuente: Elaboración propia

Lógica interna: En la lógica interna, se realiza en la transición de dos estados para que se cumpla el primer estado es realizar la acción de HO y visualmente se iluminara el segundo led azul (Figura 4.20)

Figura 4.20: Transición interna del primer sub-estado

Fuente: Elaboración propia

En la figura 4.21. se corroborará el desplazamiento de los actuadores para el primer estado interno.

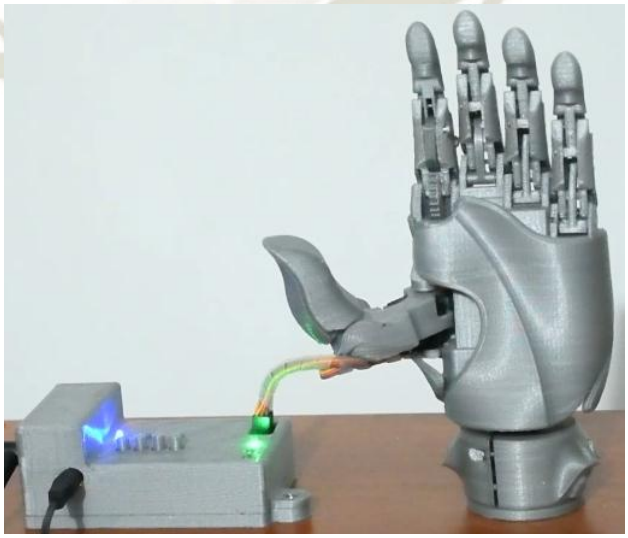


Figura 4.21: Desplazamiento de la mano del segundo estado y primer estado interno

Fuente: Elaboración propia

Para que se cumpla el segundo estado es realizar la acción de DHO y visualmente se iluminara el tercer led azul (Figura 4.22)

Figura 4.22: Transición interna del segundo sub-estado

Fuente: Elaboración propia

En la figura 4.23. se corroborará el desplazamiento de los actuadores para el segundo estado interno.

Figura 4.23: Desplazamiento de la mano del segundo estado y segundo estado interno

Fuente: Elaboración propia

c. ESTADO 2

Para cumplir este estado verificamos que el **tercer** led verde se encienda y se encuentre en el **primer** led azul también encendido, que es la condición de reposo de los servomotores.

Figura 4.24: Modulo electrónico en el tercer estado principal

Fuente: Elaboración propia

Lógica interna: En la lógica interna, se realiza en la transición de dos estados para que se cumpla el primer estado es realizar la acción de HO y visualmente se iluminara el segundo led azul (Figura 4.25)

Figura 4.25: Transición interna del primer sub-estado

Fuente: Elaboración propia

En la figura 4.26. se corroborará el desplazamiento de los actuadores para el primer estado interno.

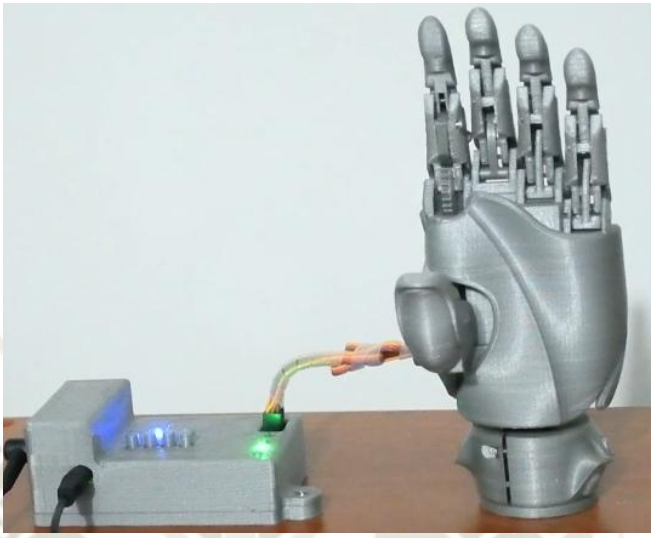


Figura 4.26: Desplazamiento de la mano del tercer estado y primer estado interno

Fuente: Elaboración propia

Para que se cumpla el segundo estado es realizar la acción de DHO y visualmente se iluminara el tercer led azul (Figura 4.27)

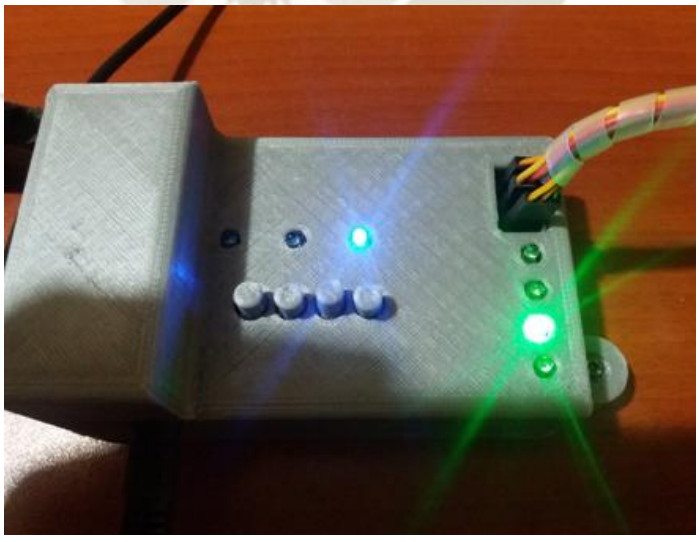


Figura 4.27: Transición interna del segundo sub-estado

Fuente: Elaboración propia

En la figura 4.28. se corroborará el desplazamiento de los actuadores para el segundo estado interno.

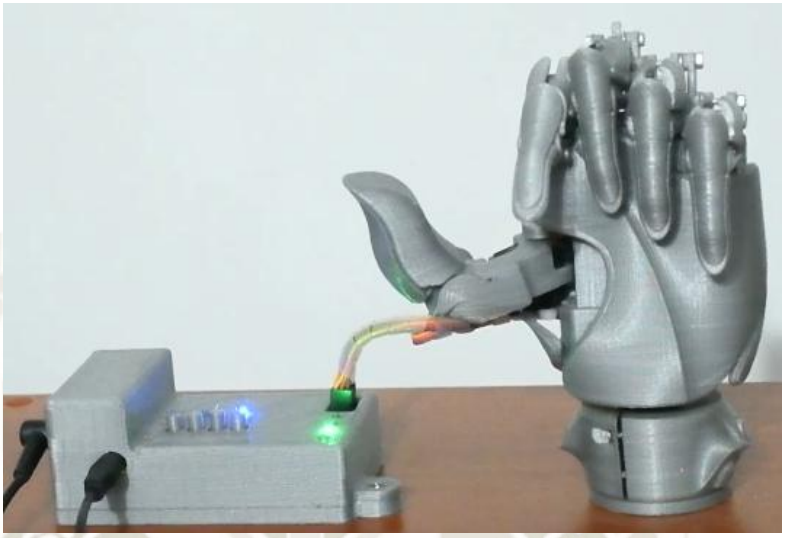


Figura 4.28: Desplazamiento de la mano del tercer estado y segundo estado interno

Fuente: Elaboración propia

d. ESTADO 3

Para cumplir este estado verificamos que el **cuarto** led verde se encienda y se encuentre en el **primer** led azul también encendido, que es la condición de reposo de los servomotores.

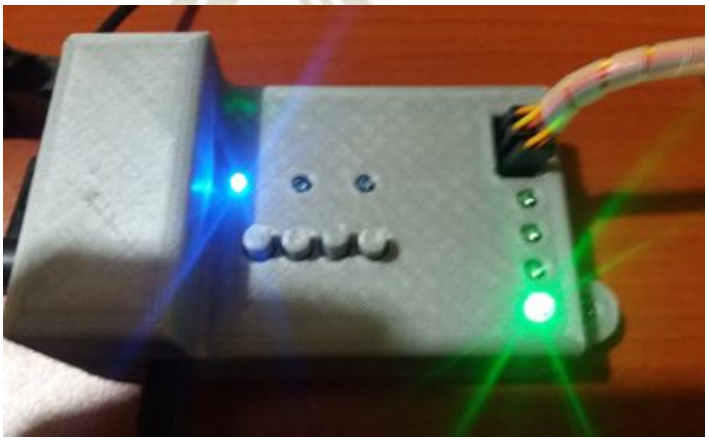


Figura 4.29: Modulo electrónico en el cuarto estado principal

Fuente: Elaboración propia

Lógica interna: En la lógica interna, se realiza en la transición de dos estados para que se cumpla el primer estado es realizar la acción de HO y visualmente se iluminara el segundo led azul (Figura 4.30)

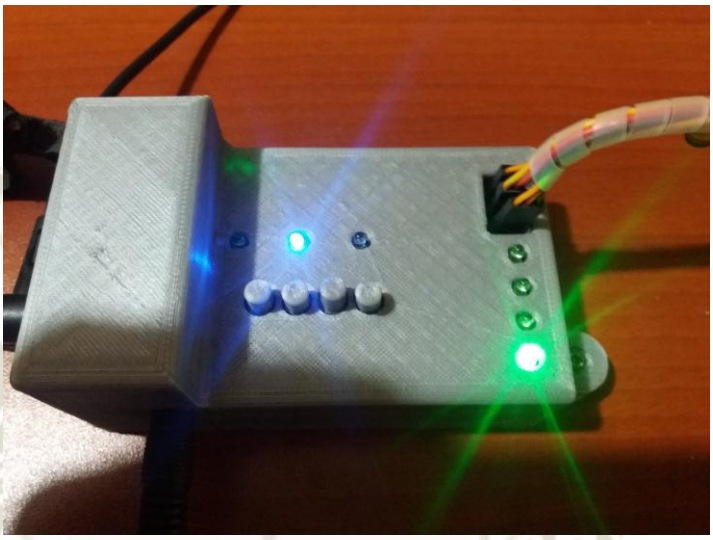


Figura 4.30: Transición interna del primer sub-estado

Fuente: Elaboración propia

En la figura 4.31. se corroborará el desplazamiento de los actuadores para el primer estado interno.

Figura 4.31: Desplazamiento de la mano del cuarto estado y primer estado interno

Fuente: Elaboración propia

Para que se cumpla el segundo estado es realizar la acción de HO y visualmente se iluminara el tercer led azul (Figura 4.32)

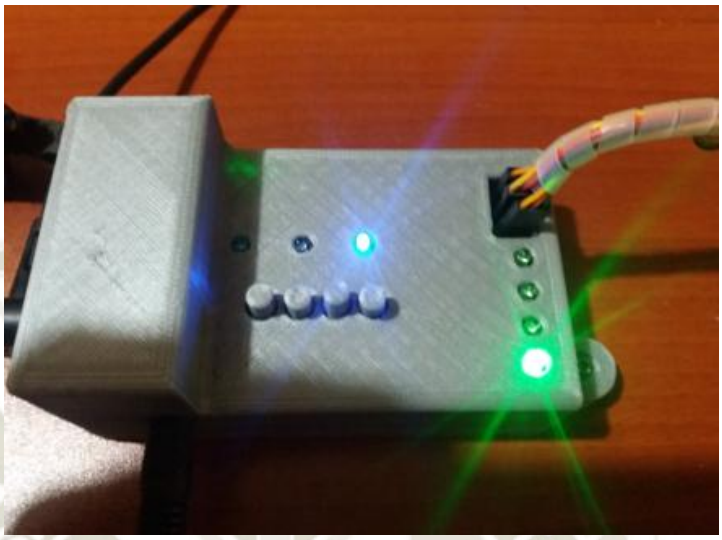


Figura 4.32: Transición interna del segundo sub-estado

Fuente: Elaboración propia

En la figura 4.33. se corroborará el desplazamiento de los actuadores para el segundo estado interno.

Figura 4.33: Desplazamiento de la mano del cuarto estado y segundo estado interno

Fuente: Elaboración propia

4.4. ETAPA 4 – Interfaz Virtual LABVIEW

Para realizar la comunicación con la interfaz gráfica de usuario se conecta un cable USB tipo A al USB tipo C del módulo electrónica donde se monitorea todos los parámetros, la explicación paso se puede encontrar el desarrollo de ingeniería sección 3.3.5.

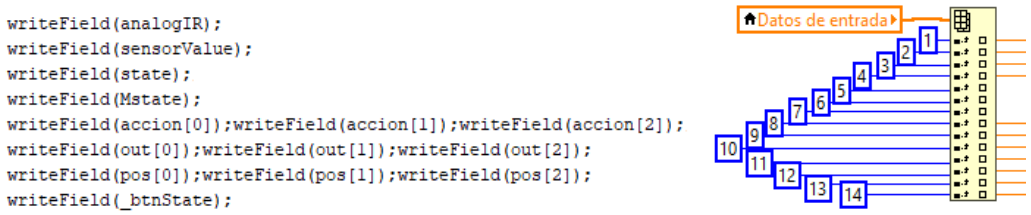


Figura 4.34: Comunicación y datos seleccionados

Fuente: Elaboración propia

Para realizar el monitoreo se requiere seleccionar el puerto serial del módulo electrónico y presionar iniciar.

Ahora explicaremos la interacción gráfica con el usuario.

En la figura 4.35 podemos ver que antes de poder ejecutar el programa se requiere seleccionar el puerto serial de la tarjeta electrónica con la que realizará la comunicación.

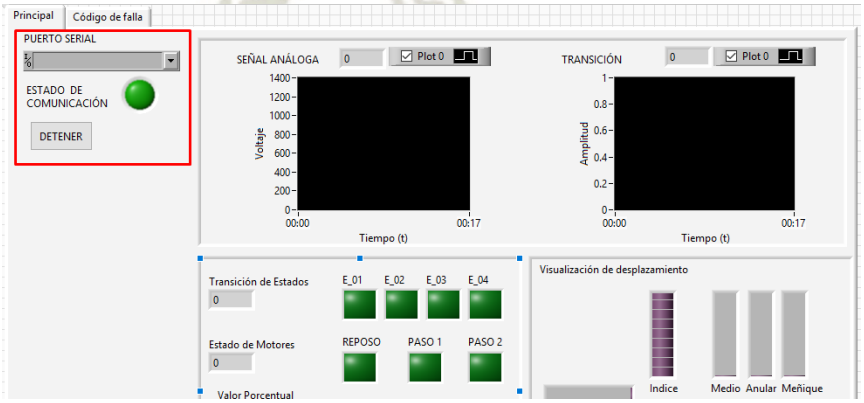


Figura 4.35. Comunicación Serial

Fuente: Elaboración propia

En la figura 4.36 logramos ver dos áreas de gráficos, siendo el de la izquierda que mostrará los parámetros análogos leídos y el de la derecha que mostrará el acondicionamiento a valores digitales.

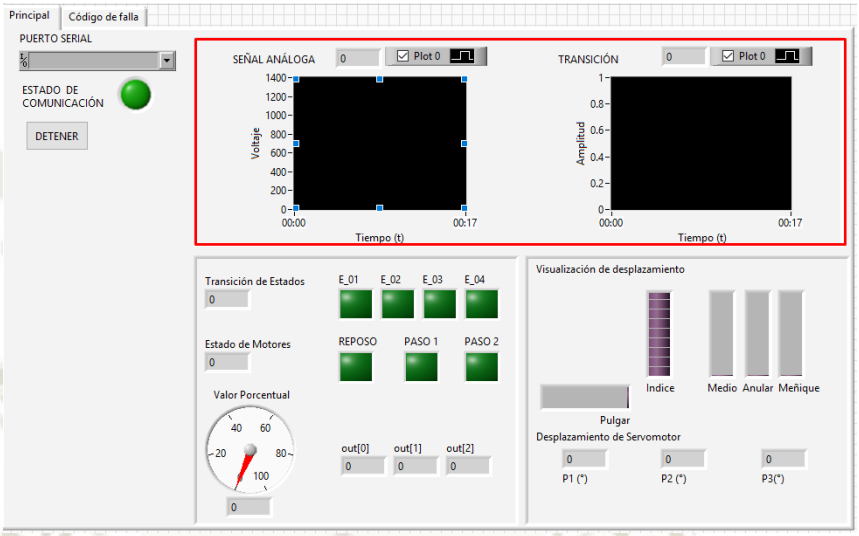


Figura 4.36. Visualización de la señal análoga

Fuente: Elaboración propia

En la figura 4.37 tendremos la transición de estados y los estados de los motores (Sub estados o transiciones internas estas explicadas en las secciones 3.2.3. y 3.3.4.).

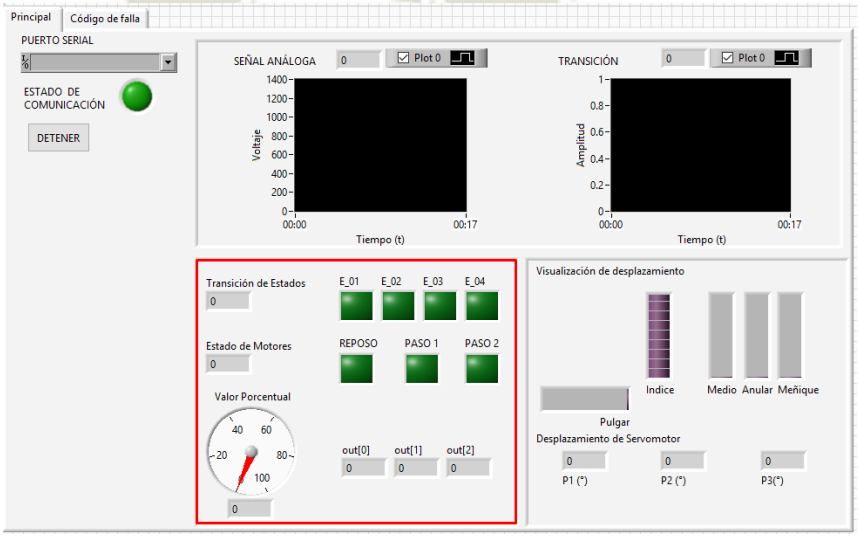


Figura 4.37. GUI en funcionamiento del sistema

Fuente: Elaboración propia

En la figura 4.38 se visualizarán los desplazamientos reales de los servomotores, siendo esta la señal de control de salida a la mano robótica y poder comprar con el sistema real.

Figura 4.38. GUI de desplazamiento de dedos

Fuente: Elaboración propia

Ya funcionando el todo el sistema, monitoreamos todos los parámetros análogos y digitales. Considerando los más importantes las transiciones de estado como los estados de los motores, la lectura análoga, digitales y las transiciones (Figura 4.39).

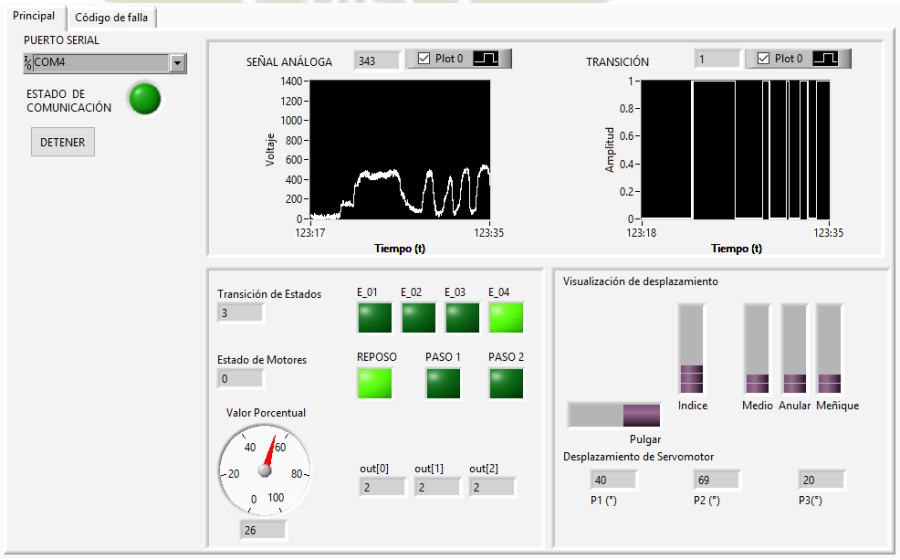


Figura 4.39. GUI en funcionamiento del sistema

Fuente: Elaboración propia

CONCLUSIONES

- Se logró construir una mano robótica con tecnología de impresión 3D con el material de PLA y la implementación del control por optomiografía al crear la lógica de control para que pueda recibir la información del sensor IR, para luego procesarla y ejecutar la flexión con extensión de los dedos enviando señales de control a los servomotores (Figura 4.13).
- Al finalizar la impresión de la mano robótica con el material PLA, se logró desarrollar y construir una mano robótica que sea robusto, seguro, ligero y transportable debido al material utilizado y usar mecanismos de anclaje y unión de cada pieza (Figura 4.13).
- Se logró diseñar un control por optomiografía (Figura 3.3.) y funcione adecuadamente al realizar el acondicionamiento de señal que se creó dos tipos de diagrama de estados, siendo el primero de cuatro estados y el ultimo de dos estados que logran manejar de forma interna en cada uno de los estados del primero por señales de transición.
- Se elaboró una interfaz gráfica con la cual puede validar el comportamiento de la mano robótica HACKBERRY, con la que se pudo lograr el monitoreo de los parámetros recibidos y enviados por el microcontrolador a través del protocolo de comunicación RS232 (Figura 4.35).
- Se elaboró una banda con el sensor IR que logró posicionar el sensor a una distancia de 4 mm de la piel puedo lograr la correcta detección de la contracción muscular próximo al musculo pronador (Figura 4.2).
- Se validó que los efectores funcionaron de forma correcta durante la ejecución del proyecto y estos trabajen dentro de su rango de operación. Al corroborar las 8 posibles condiciones debido a los 3 servomotores que trabajan independiente.

Siendo básicamente las condiciones de abrir, cerrar y detener la posición desplazada (Capitulo 4 en la etapa 4.3.).

RECOMENDACIONES

- Durante el proceso de calibración es muy importante realizar la acción de mantener-soltar (HO, Hold-Open) para poder establecer los rangos máximos y mínimos de lectura.
- La lógica de desplazamiento de lo servomotores se puede modificar en la matriz de 3x8, según sea conveniente.
- El beneficio que presenta el programa diseñado es su estructura modular dando facilidad de poder ampliar y aumentar los grados de libertad, como poder aplicarlo en cualquier mano robótica que uno desee.
- El programa diseñado para el monitoreo (LABVIEW-ARDUINO_IDE), presenta autonomía en la recepción de datos, también dando facilidad a la lectura de estos y conectar al sistema de la mano en el momento que el usuario desee.

BIBLIOGRAFÍA

1. BRUNO SOSPEDRA GRIÑO, (2015). Diseño mecánico de prótesis de mano multidedo antropomórfica infractuada, (pp.35).
2. Artículo de prótesis de la revista de la UNAM. http://www.revista.unam.mx/vol.6/num1/art01/art01-2d.htm.
3. HAMED HAMID MUHAMMED, JAMMALAMADAKA RAGHAVENDRA (2015) Optomyography (OMG): A Novel Technique for the Detection of Muscle Surface Displacement Using Photoelectric Sensors.
4. Mano robótica DEXTRUS. https://bluebadgestyle.com.
5. Mano comercial BIONIC. http://tiendaonline.juanbravo.com.
6. Teoría de máquinas de estado finito http://ai-depot.com/FiniteStateMachines/FSM-Background.html
7. THE UNIVERSITY OF TEXAS AT AUSTIN. [2014]. Embedded Systems – Shape the World. https://courses.edx.org/courses/UTAustinX/UT.6.01x/1T2014/
8. Reportaje de cuerpo biónico, ya no es ciencia ficción. https://nationalpost.com.
9. FINCH, mano robótica alternativa. http://finch-hand.jp/index_en.html.
10. JOSÉ VALDEIGLESIAS, (2015). Diseño y construcción de una mano protésica. Universidad Católica de Santa María (pp. 83).
11. ENEGIA SOFTWARE, programación de código abierto. http://energia.nu/
12. Robótica y prótesis inteligentes. http://www.revista.unam.mx/vol.6/num1/art01/art01_enero.pdf.
13. FLOYD, novena edición. Fundamentos digitales. Pearson International Edition. (pp. 311).
14. Teoría de opto-acopladores https://www.luisllamas.es/arduino-optoacoplador

15. VILARINO, MARTIN, (2013) Enhancing the Control of Upper Limb Myoelectric Prostheses Using Radio Frequency Identification. Universidad Johns Hopkins.
16. Teoría de máquinas de estado finito. http://ai-depot.com/FiniteStateMachines/FSM-Background.html
17. KRISTIAN P. NIELSEN, LU ZHAO, JAKOB J. STAMNES, KNUT STAMNES, JOHAN MOAN, (2008) The optics of human skin: Aspects important for human health (pp. 5)
18. Teoría de opto-acopladores. https://www.luisllamas.es/arduino-optoacoplador.
19. Conexionado a la tarjeta embebida http://energia.nu/pinmaps/#_32_bit_tiva_c.

ANEXOS

A.1. DISEÑO ELECTRÓNICO

A.1.1. Diseño electrónico del módulo electrónico

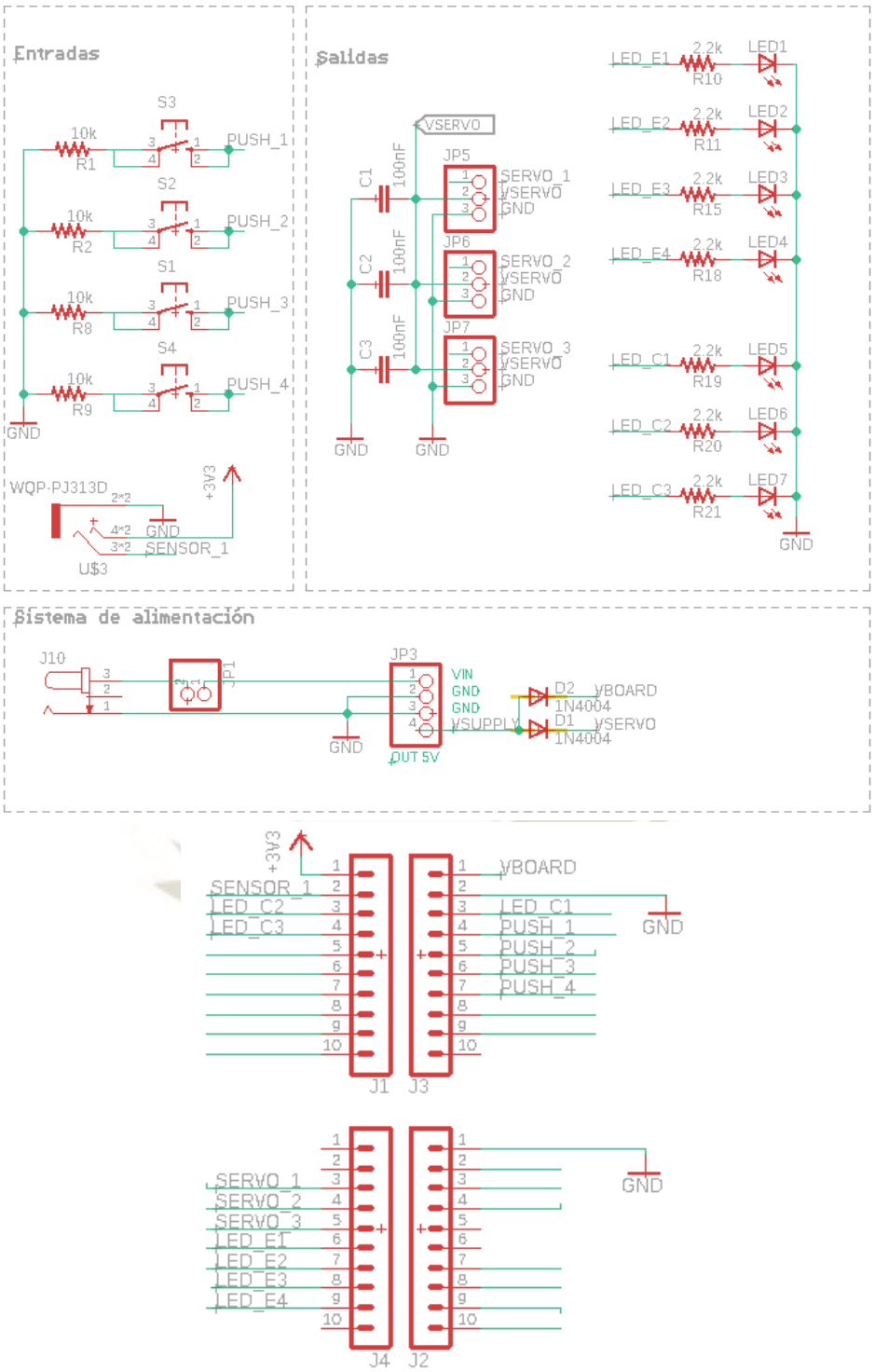


Figura A.1.1. *Diagrama electrónico del módulo electrónica*

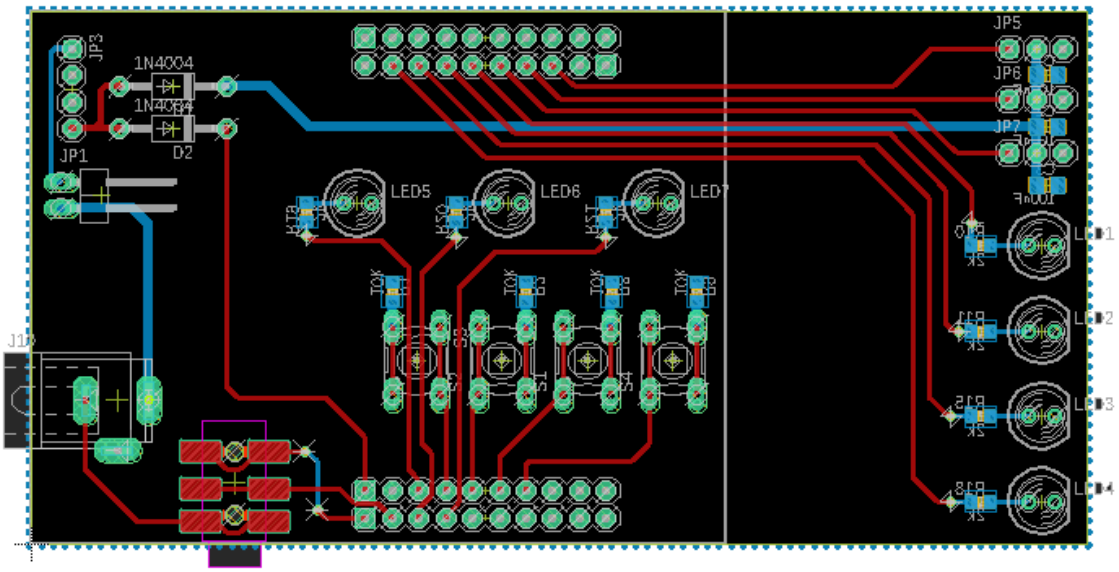


Figura A.1.2. *Desarrollo de la tarjeta por EAGLE*

A.1.2. Diseño electrónico del encapsulado IR

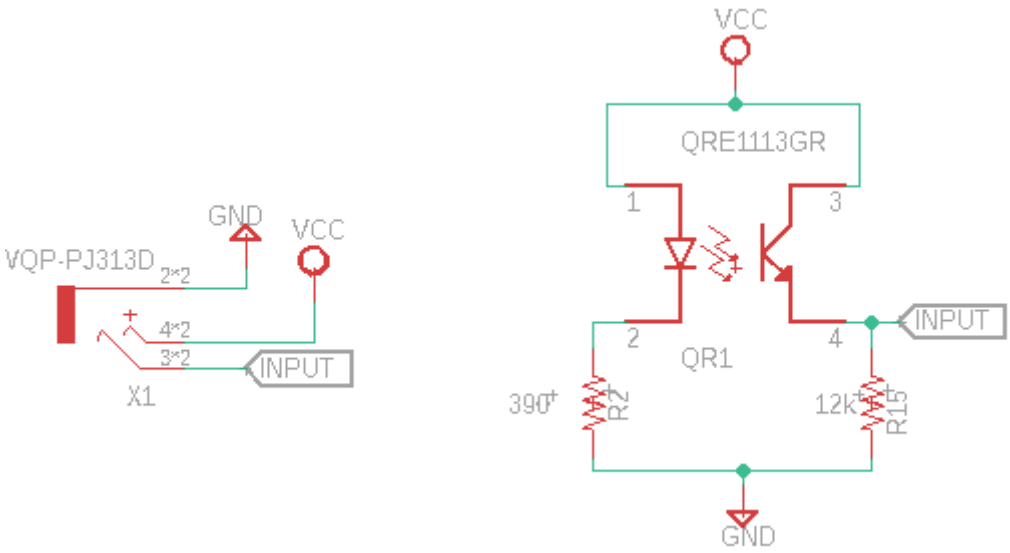


Figura A.1.3. *Diagrama electrónico la tarjeta del encapsulado*

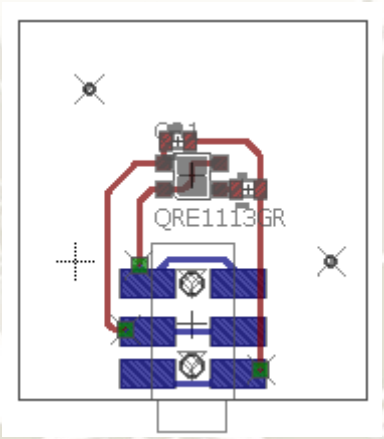


Figura A.1.4. *Desarrollo de la tarjeta por EAGLE*

A.2. PROGRAMACIÓN

A.2.1. Programación en C para el microcontrolador

```
#include <Servo.h>
Servo myservo_1;
Servo myservo_2;
Servo myservo_3;

const int PinServo[3]={PD_6, PC_7, PC_6};          // Asignación de Puertos
const int pushbutton[4]={PD_1, PD_2, PD_3, PE_1};  // Asignación de Puertos
int buttonState[4] = {0, 0, 0, 0};                 // Vector de Variable - Variable para lectura del
estado de pushbutton

const int Led_Servos[3]={PD_0, PB_0, PB_1};        // Asignación de Puertos
const int Led_Estados[4]={PC_5, PC_4,PB_3, PF_3}; // Asignación de Puertos

int prevstate = 6;
int prevMstate = 6;

int val, increment = 1, updateInterval = 20;    // variable to read the value from the analog pin
volatile unsigned long lastUpdate;
int pos[3]={0,0,0};

int minlimit[3]={20,5,20};                      // Limites Mínimos del Servomotor
int maxlimit[3]={90,90,90};                     // Límites Máximos del Servomotor

int state=0;                                    // Variables del estado Máquina
char input;
int Mstate=0;

int analogIR = 0;
volatile unsigned long previousMillis = 0;       // Envío de información a LABVIEW
const long interval = 5;                         // Periodo de envío de trama al LABVIEW
//-------------------------------------------------------------------------------------------
// Inicio - Análoga | Variables Lectura análoga
//-------------------------------------------------------------------------------------------

   // variables:
   int sensorValue = 0;         // the sensor value


   // ------> logica Digital
   int clicks;
   boolean depressed
   long multiclickTime;
   long longClickTime;

   int _pin;
   boolean _activeHigh;
   boolean _btnState;
   boolean _lastState;
   int _clickCount;
   long _lastBounceTime;
```

```
   boolean activeType = LOW;
const int buttonPin = PB_5;         // valor analógico

   int in;

const int ledPin = PF_1;

//---------------------------------------------------------------------------------------------
// Fin    - Análoga | Variables Lectura análoga
//---------------------------------------------------------------------------------------------
// Inicio - Estado Máquina | Variables
//---------------------------------------------------------------------------------------------

int accion1,
   accion2,
   accion3,
   accion[3];

// Variables de salida
int out[3];

const int fingerpos [8][3] = {{1,0,0}, // State 0 pulgar/index/otros
                              {0,1,0}, // State 1
                              {1,1,0}, // State 2
                              {0,1,1}, // State 3

                              {1,1,1}, // State 0
                              {1,0,1}, // State 1
                              {0,0,1}, // State 2
                              {0,0,0}}; // State 3

//---------------------------------------------------------------------------------------------
// Fin    - Estado Máquina | Variables
//---------------------------------------------------------------------------------------------

void servoControl(int a,int b,int c)
{
  int control[3]= {a,b,c};

   if((millis() - lastUpdate) > updateInterval)
  {
    for (int i=0; i < 3; i++){

    if ( control[i] == 0  ) { pos[i] += increment;      }
    if ( control[i] == 1 ) { pos[i] -= increment;      }
    if ( control[i] == 2  )  {pos[i] = pos[i];}
    //=== Saturacion  =========
    if (pos[i] <= minlimit[i])   {pos[i]=minlimit[i];}
    if (pos[i] >= maxlimit[i]) {pos[i]=maxlimit[i];}
    //==========================
    }
    lastUpdate = millis();
  }
}
```

```
// la resolucion de salida es de un maximo y mínimo de: -99999.99 a 999999.99 como valor de salida.
// para modificar es necesario reacondicionar la señal en el programa de labview.
//.....
//Funcion principal
void writeField(int val)
{
  String sendd=String(val);
  Serial.print(F(",")); // start byte
  //Serial.print(sendd.length());
  Serial.print(sendd);
 }


void setup()
{
  Serial.begin(115200);

// Setup el timer del IR (Todo en milisegundos / ms)
// (Estarán colocadas por defecto, pero se puede modificar según sea conveniente)
  _pin          = buttonPin;
  _activeHigh    = activeType;
  _btnState      = !_activeHigh;  // Estado logico inicial es activo alto del sensor IR
  _lastState     = _btnState;
  _clickCount    = 0;
  clicks        = 0;
  depressed      = 0;
  _lastBounceTime= 0;
  debounceTime   = 20;           // Rebote timer en ms
  multiclickTime = 350;          // Tiempo limite para el multiPulso
  longClickTime  = 700;          // Tiempo registro hasta el "Largo" pulso

// Asignación de servos y variables de inicio
  myservo_1.attach(PinServo[0]);
  myservo_2.attach(PinServo[1]);
  myservo_3.attach(PinServo[2]);

  myservo_1.write(minlimit[0]);
  myservo_2.write(minlimit[1]);
  myservo_3.write(minlimit[2]);

  pinMode(ledPin,OUTPUT);

  pinMode(pushbutton[0],INPUT_PULLUP);
  pinMode(pushbutton[1],INPUT_PULLUP);
  pinMode(pushbutton[2],INPUT_PULLUP);
  pinMode(pushbutton[3],INPUT_PULLUP);

  pinMode(Led_Estados[0],OUTPUT);
  pinMode(Led_Estados[1],OUTPUT);
  pinMode(Led_Estados[2],OUTPUT);
  pinMode(Led_Estados[3],OUTPUT);
```

```
  pinMode(Led_Servos[0],OUTPUT);
  pinMode(Led_Servos[1],OUTPUT);
  pinMode(Led_Servos[2],OUTPUT);

  pinMode(buttonPin,INPUT); // Pin analógico
/*
// Bucle que se ejecutará mientas no se active el pulsador
buttonState[0] = digitalRead(pushbutton[0]);
do
{
  buttonState[0] = digitalRead(pushbutton[0]);

} while (buttonState[0] == HIGH);

lastUpdate = millis();

// Calibración del sensor IR - Duración 5seg
Serial.print(F("Inicio de la Calibracion "));

while (millis() - lastUpdate < 5000) {

    sensorValue = analogRead(buttonPin);
   // record the maximum sensor value
    if (sensorValue > sensorMax) { sensorMax = sensorValue; }

    // record the minimum sensor value
    if (sensorValue < sensorMin) { sensorMin = sensorValue; }
  }

Serial.println(F("Fin de la Calibracion "));
delay(1000);

*/

}
void loop()
{
  volatile long now = (volatile long)millis();      // Tiempo actual dado por temporizador del
microcontrolador

  //--------------------------------------------------------------------------------------------
  // Inicio    - Lectura y procesamiento de Sensor IR
  //--------------------------------------------------------------------------------------------

  // Lectura de la variable analoga
  sensorValue = analogRead(_pin); /*  Serial.print(" Sensor analog: ");
Serial.print(sensorValue);*/
  analogIR = sensorValue;  // Valor real
  // Aplicado a los variables calibradas en el Setup
  sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 100);
  sensorValue = constrain(sensorValue, 0, 100);

  // En caso que el valor del sensor se encuentre fuera de rango verifique la calibración

  if (sensorValue > 50 )
```

```
  {
    digitalWrite(ledPin, LOW);
    _btnState = LOW;  // current appearant button state

  }
  else
  {
   digitalWrite(ledPin, HIGH);
    _btnState = HIGH;  // current appearant button state
  }

  // Make the button logic active-high in code
  if (!_activeHigh) _btnState = !_btnState;
  // If the switch changed, due to noise or a button press, reset the debounce timer
  if (_btnState != _lastState) _lastBounceTime = now;
  // debounce the button (Check if a stable, changed state has occured)
  if (now - _lastBounceTime > debounceTime && _btnState != depressed)
  {
    depressed = _btnState;
    if (depressed) _clickCount++;
  }
  // If the button released state is stable, report nr of clicks and start new cycle
  if (!depressed && (now - _lastBounceTime) > multiclickTime)
  {
    // positive count for released buttons
    clicks = _clickCount;
    _clickCount = 0;
  }
  // Check for "long click"
  if (depressed && (now - _lastBounceTime > longClickTime))
  {
    // negative count for long clicks
    clicks = 0 - _clickCount;
    _clickCount = 0;
  }

  _lastState = _btnState;

  in = clicks;

//  Serial.println(in);

//-----------------------------------------------------------------------------------------
// Fin    - Lectura y procesamiento de Sensor IR
//-----------------------------------------------------------------------------------------
// Inicio - Activacion de LEDs
if (state != prevstate)
{
digitalWrite(Led_Estados[state],HIGH);
digitalWrite(Led_Estados[prevstate],LOW);
prevstate=state;
}

if (Mstate != prevMstate)
{
```

```
digitalWrite(Led_Servos[Mstate],HIGH);
digitalWrite(Led_Servos[prevMstate],LOW);
prevMstate=Mstate;
}
// Fin    - Activacion de LEDs

// Estados lógicos

 if (in == 1) {
  input = 'C';
 }
 else if (in == 2) {
  input = 'D';
 }
 else if (in == 3) {
  input = 'T';
 }

 else if (in == 0) {
  input = 0;
 }

 else if (in == -1) {
  input = 'H';
 }
 else if (in == -2) {
  input = 'J';
 }


 // Estado 0
 if (state == 0) {

  if (input == 0)
  {
   state = 0;
  }
  else if ( input == 'C')
  {
   state = 1;
  }
  else if ( input == 'D')
  {
   state = 2;
  }
  else if ( input == 'T')
  {
   state = 3;
  }
 }
 // Estado 1
 else if (state == 1) {

  if (input == 0)
  {
```

```
    state = 1;
   }
   else if ( input == 'C')
   {
    state = 0;
   }
   else if ( input == 'D')
   {
    state = 2;
   }
   else if ( input == 'T') //HOLD ON
   {
    state = 3;
   }
  }
  // Estado 2
  else if (state == 2) {

   if (input == 0)
   {
    state = 2;
   }

   else if ( input == 'C')
   {
    state = 0;
   }
   else if ( input == 'D')
   {
    state = 1;
   }
   else if ( input == 'T') //HOLD ON
   {
    state = 3;
   }
  }
  // Estado 3
  else if (state == 3) {

   if (input == 0)
   {
    state = 3;
   }

   if ( input == 'C')
   {
    state = 0;
   }
   else if ( input == 'D')
   {
    state = 1;
   }
   else if ( input == 'T') //HOLD ON
   {
    state = 2;
```

```
    }

  }

  else  {
    // No colocar ningún código aquí
  }

//-------------------------------------------------------------------------------------------
//ACCION DE SALIDA A MOTOR
//-------------------------------------------------------------------------------------------

  // Estado 0 - reposo
  if (Mstate == 0) {

    if (state == 0 || state == 1 || state == 2 || state == 3) // Aseguro condición de reposo.
    {
      accion1 =  2; // condición de reposo en la función servoControl()
      accion2 =  2; // condición de reposo en la función servoControl()
      accion3 =  2; // condición de reposo en la función servoControl()
    }

    if (input == 'H')
    {
      Mstate = 1;
    }
    else if (input == 'J')
    {
      Mstate = 2;
    }
  }

  // Estado 1
  else if (Mstate == 1) {

    if (state == 0)
    {
      accion1 =  fingerpos[0][0];
      accion2 =  fingerpos[0][1];
      accion3 =  fingerpos[0][2];
    }
    else if (state == 1)
    {
      accion1 =  fingerpos[1][0];
      accion2 =  fingerpos[1][1];
      accion3 =  fingerpos[1][2];
    }
    else if (state == 2)
    {
      accion1 =  fingerpos[2][0];
      accion2 =  fingerpos[2][1];
      accion3 =  fingerpos[2][2];
    }
    else if (state == 3)
    {
```

```
    accion1 =  fingerpos[3][0];
    accion2 =  fingerpos[3][1];
    accion3 =  fingerpos[3][2];
   }

   if (input == 0 && _btnState == 0) // pullup in Aseguro condición
   {
    Mstate = 0;
   }

 }
 // Estado 2
 else if (Mstate == 2 ) {

   if (state == 0)
   {
    accion1 =  fingerpos[4][0];
    accion2 =  fingerpos[4][1];
    accion3 =  fingerpos[4][2];
   }
   else if (state == 1)
   {
    accion1 =  fingerpos[5][0];
    accion2 =  fingerpos[5][1];
    accion3 =  fingerpos[5][2];
   }
   else if (state == 2)
   {
    accion1 =  fingerpos[6][0];
    accion2 =  fingerpos[6][1];
    accion3 =  fingerpos[6][2];
   }
   else if (state == 3)
   {
    accion1 =  fingerpos[7][0];
    accion2 =  fingerpos[7][1];
    accion3 =  fingerpos[7][2];
   }

   //  servoControl(2);
   if (input == 0 && _btnState == 0) // pullup in Aseguro condición
   {
    Mstate = 0;
   }
 }


//-------------------------------------------------------------------------------------------
//================= Señal de salida al actuador
==================================
//-------------------------------------------------------------------------------------------
  accion[0] = accion1;
  accion[1] = accion2;
  accion[2] = accion3;
```

```
 // LÓGICA DE SALIDA 02 - SIN FSR

for (int i = 0; i < 3; i++) {
   out[i] = accion[i];
  }

  // función de cálculo de desplazamiento de servos
  servoControl(out[0], out[1], out[2]);

  // colocar la posición según el orden escogido
  myservo_1.write(pos[0]);
  myservo_2.write(pos[1]);
  myservo_3.write(pos[2]);

// cada cierto intervalo de tiempo se envía los datos por el puerto serial
  volatile unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    // Envio de señales en string
    Serial.print("I");
    writeField(analogIR);// lectura real del sensor
    writeField(sensorValue); // luego de la calibración en el setup del código
    writeField(state);// Estado lógico
    writeField(Mstate);// Estado Motor (posee 3 estados)
    writeField(accion[0]);writeField(accion[1]);writeField(accion[2]);// Acción para los motores
Boleana
    writeField(out[0]);writeField(out[1]);writeField(out[2]); // lógica real de salida al
reacondiconar los datos por condiciones de visalizacion
    writeField(pos[0]);writeField(pos[1]);writeField(pos[2]); // out 3 datos de posición de cada
servomotor
    writeField(_btnState);// transición digital
    writeField(sensorMin);// lectura real del sensor
    writeField(sensorMax); // luego de la calibración en el setup del código
    Serial.println("");
  }

     // */
}
```

A.2.2. Programación Grafica para LABVIEW

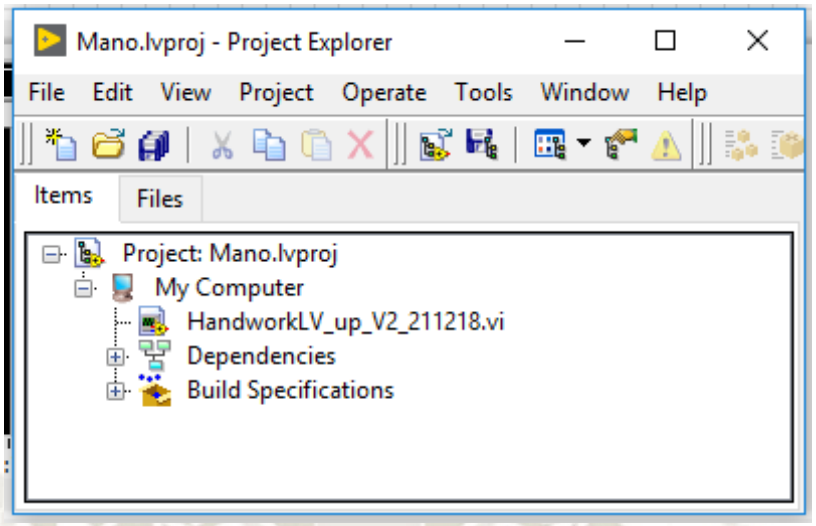


Figura A.2.1. *Proyecto mano en LABVIEW*

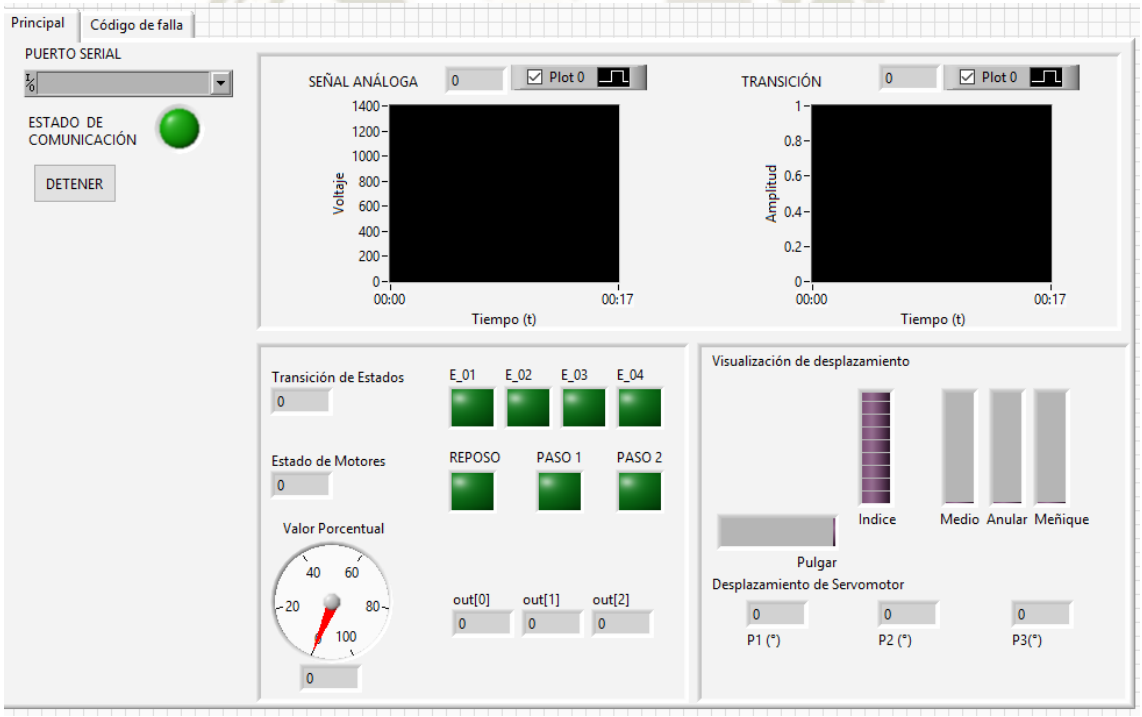


Figura A.2.2. *Interfaz Gráfica*

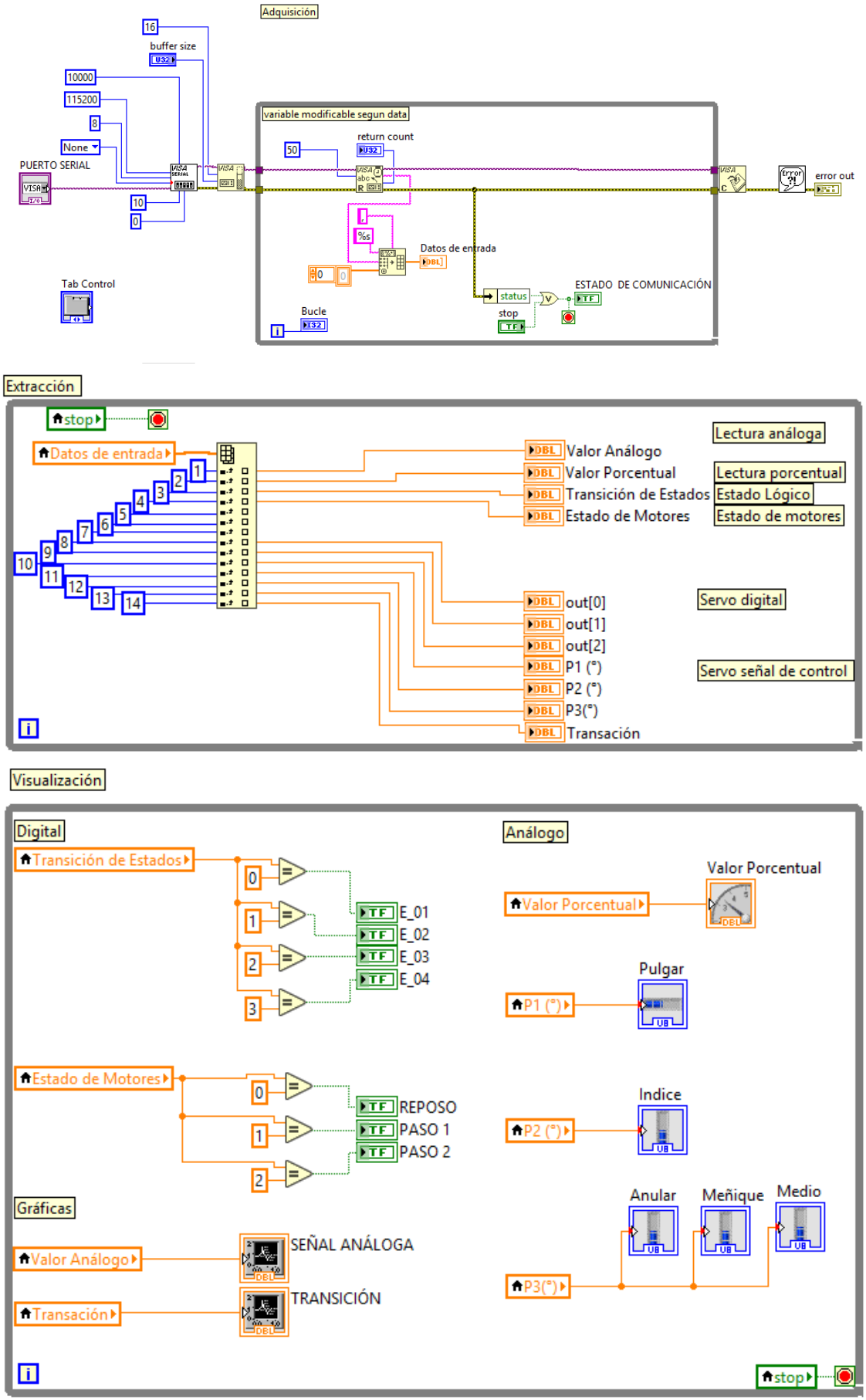


Figura A.2.3. *Programación Grafica en LABVIEW*

A.3. MAPA DE PINES DEL LAUNCHPAD TIVA C

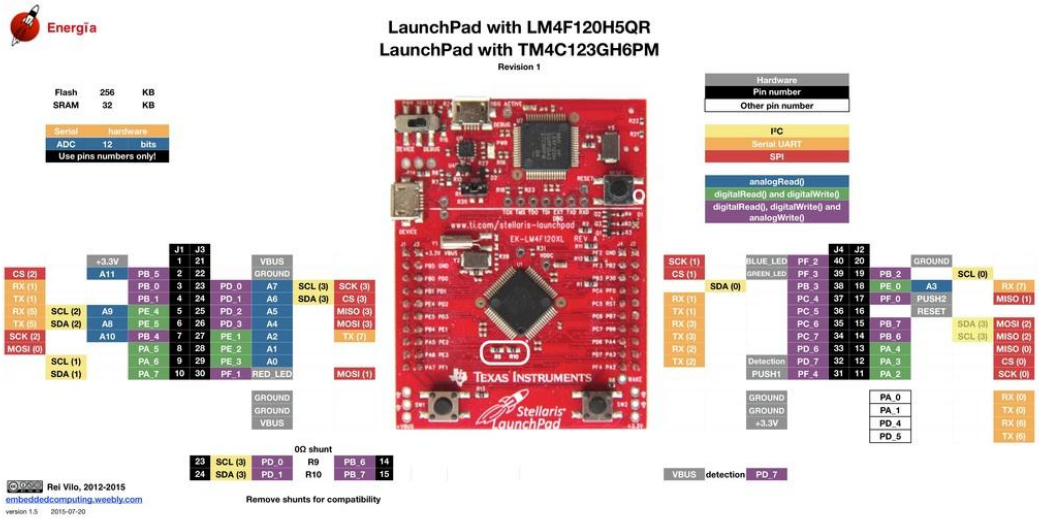


Figura A.3.1. *VISTA SUPERIOR*

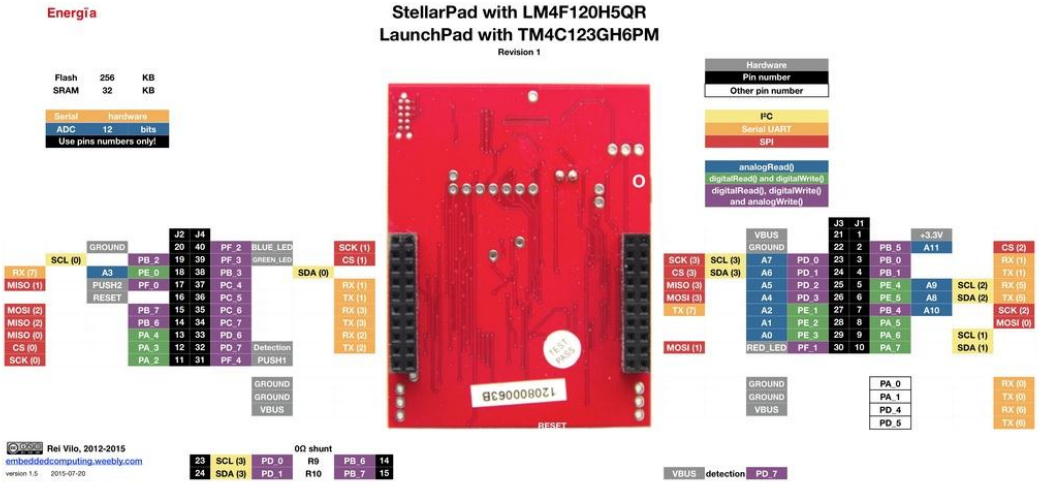


Figura A.3.2. *VISTA INFERIOR*

A.4. HOJA DE DATOS DE QRE1113

FAIRCHILD SEMICONDUCTOR®

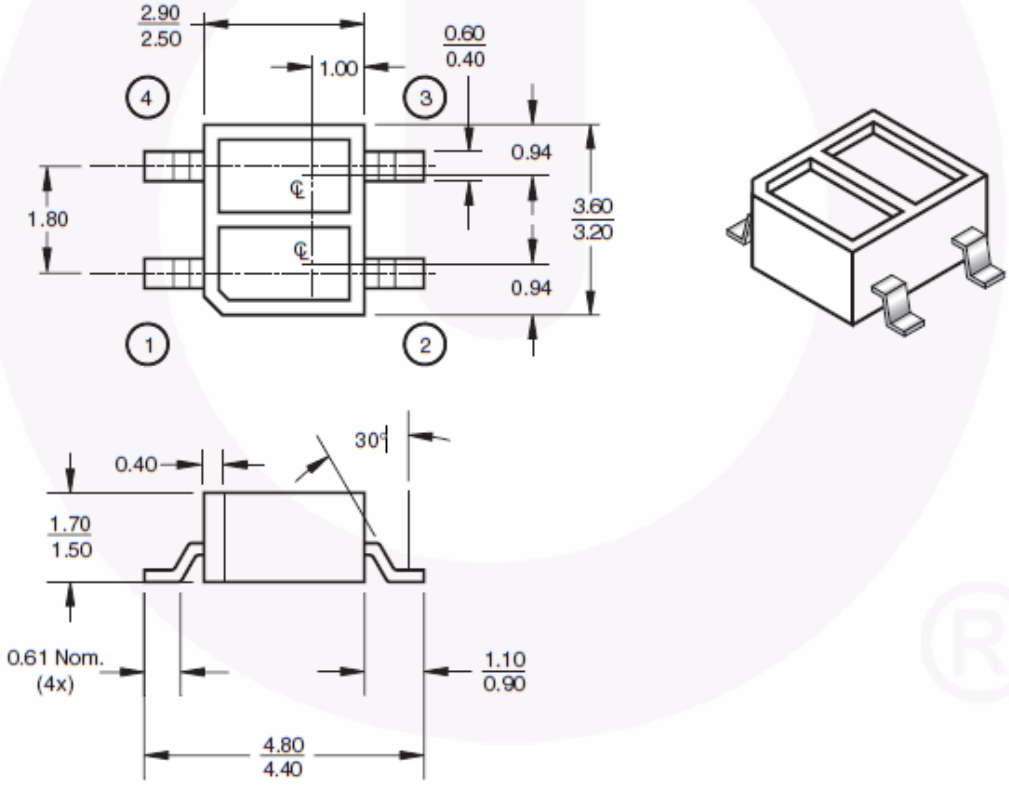
September 2009

QRE1113, QRE1113GR
Minature Reflective Object Sensor

Features

- Phototransistor output
- No contact surface sensing
- Miniature package
- Lead form style: Gull Wing
- Two leadform options: Through hole (QRE1113)
  SMT gullwing (QRE1113GR)
- Two packaging options: Tube (QRE1113)
  Tape and reel (QRE1113GR)

QRE1113GR Package Dimensions



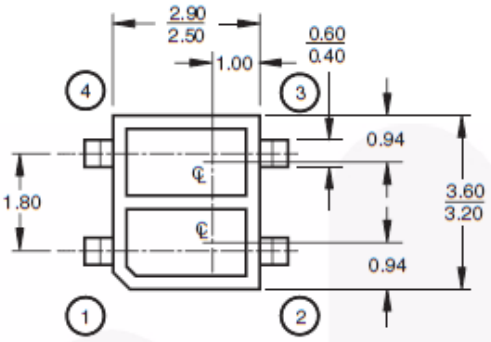
Notes:

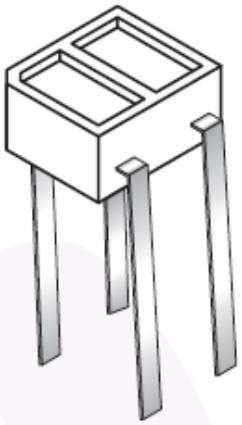
1. Dimensions for all drawings are in millimeters.
2. Tolerance of ±0.15mm on all non-nominal dimensions

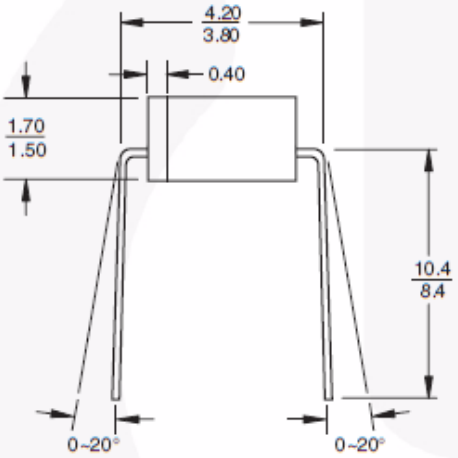
QRE1113, QRE1113GR Rev. 1.6.0

www.fairchildsemi.com

QRE1113 Package Dimensions



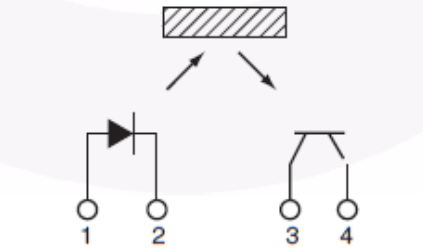




Notes:

1. Dimensions for all drawings are in millimeters.
2. Tolerance of ±0.15mm on all non-nominal dimensions

Schematic



Pin 1: Anode
Pin 2: Cathode

Pin 3: Collector
Pin 4: Emitter

Absolute Maximum Ratings (T_A = 25°C unless otherwise specified)

Stresses exceeding the absolute maximum ratings may damage the device. The device may not function or be operable above the recommended operating conditions and stressing the parts to these levels is not recommended. In addition, extended exposure to stresses above the recommended operating conditions may affect device reliability. The absolute maximum ratings are stress ratings only.

Symbol	Parameter	Rating	Units
T_{OPR}	Operating Temperature	-40 to +85	°C
T_{STG}	Storage Temperature	-40 to +90	°C
T_{SOL-I}	Soldering Temperature (Iron)[2,3,4]	240 for 5 sec	°C
T_{SOL-F}	Soldering Temperature (Flow)[2,3]	260 for 10 sec	°C
EMITTER			
I_F	Continuous Forward Current	50	mA
V_R	Reverse Voltage	5	V
I_{FP}	Peak Forward Current[5]	1	A
P_D	Power Dissipation[1]	75	mW
SENSOR			
V_{CEO}	Collector-Emitter Voltage	30	V
V_{ECO}	Emitter-Collector Voltage	5	V
I_C	Collector Current	20	mA
P_D	Power Dissipation[1]	50	mW

Electrical/Optical Characteristics (T_A = 25°C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Units
INPUT DIODE						
V_F	Forward Voltage	I_F = 20mA		1.2	1.6	V
I_R	Reverse Leakage Current	V_R = 5V			10	µA
λ_{PE}	Peak Emission Wavelength	I_F = 20mA		940		nm
OUTPUT TRANSISTOR						
I_D	Collector-Emitter Dark Current	I_F = 0mA, V_{CE} = 20V			100	nA
COUPLED						
$I_{C(ON)}$	On-State Collector Current	I_F = 20mA, V_{CE} = 5V[6]	0.10	0.40		mA
I_{CX}	Cross-Talk Collector Current	I_F = 20mA, V_{CE} = 5V[7]			1	µA
$V_{CE\ (SAT)}$	Saturation Voltage				0.3	V
t_r	Rise Time	V_{CC} = 5V, $I_{C(ON)}$ = 100µA, R_L = 1kΩ		20		µs
t_f	Fall Time			20		

Notes:

1. Derate power dissipation linearly 1.00mW/°C above 25°C.
2. RMA flux is recommended.
3. Methanol or isopropyl alcohols are recommended as cleaning agents.
4. Soldering iron 1/16" (1.6mm) from housing.
5. Pulse conditions: tp = 100µs; T = 10ms.
6. Measured using an aluminum alloy mirror at d = 1mm.
7. No reflective surface at close proximity.

Typical Performance Curves

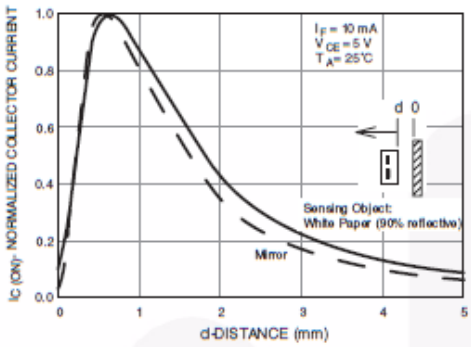


Fig. 1 Normalized Collector Current vs. Distance between device and reflector

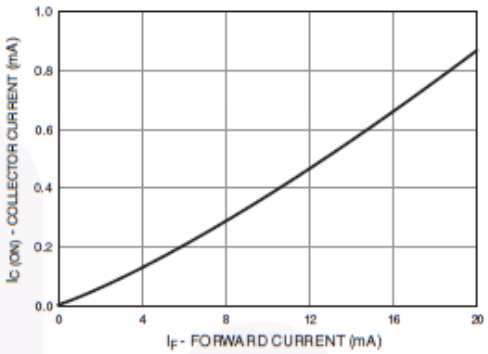


Fig. 2 Collector Current vs. Forward Current

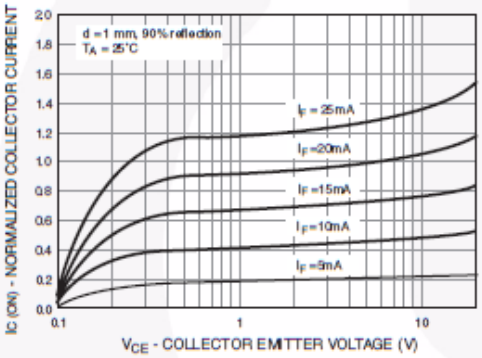


Fig. 3 Normalized Collector Current vs. Collector to Emitter Voltage

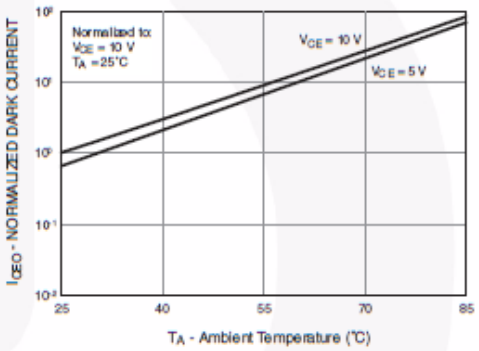


Fig. 4 Collector Emitter Dark Current (Normalized) vs. Ambient Temperature

Typical Performance Curves (Continued)

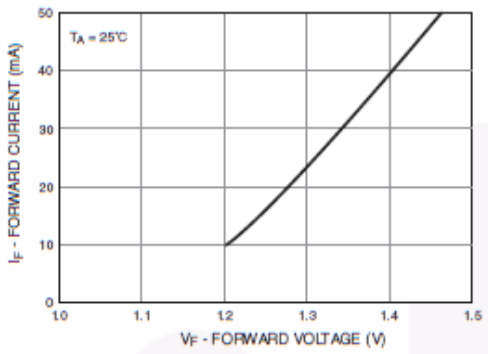


Fig. 6 Forward Current vs. Forward Voltage

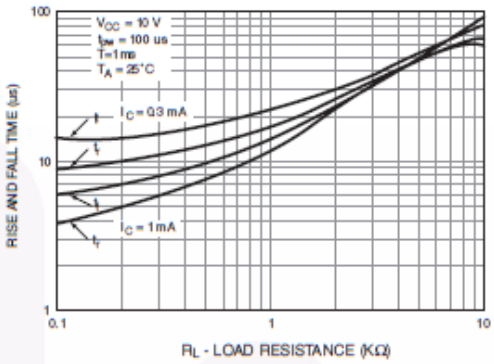


Fig. 7 Rise and Fall Time vs. Load Resistance

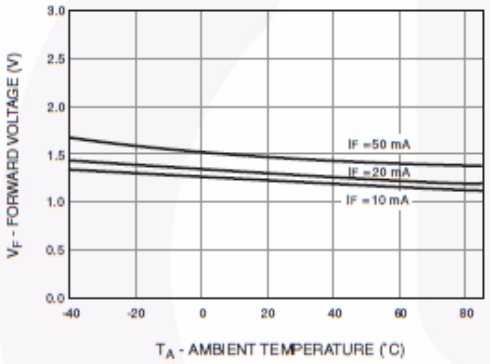


Fig. 8 Forward Voltage vs. Ambient Temperature

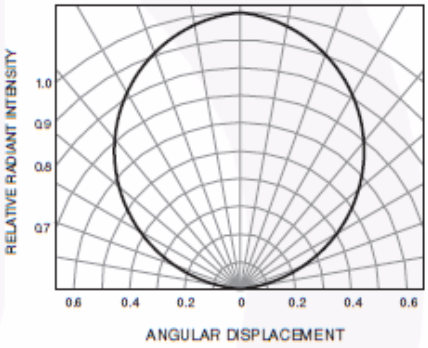
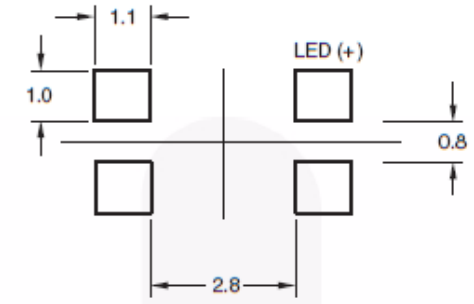


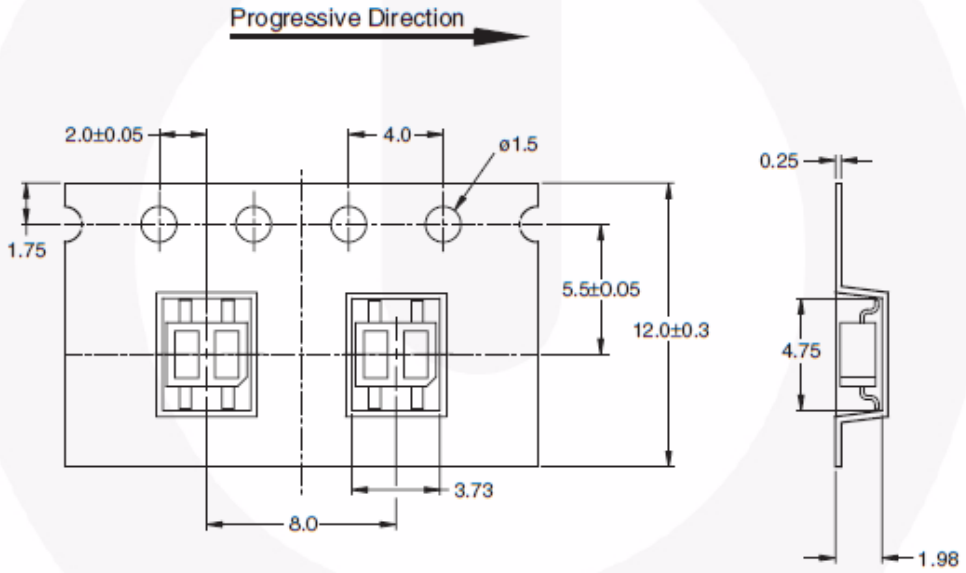
Fig. 8 Radiation Diagram

Recommended Solder Screen Pattern for GR option (for reference only)



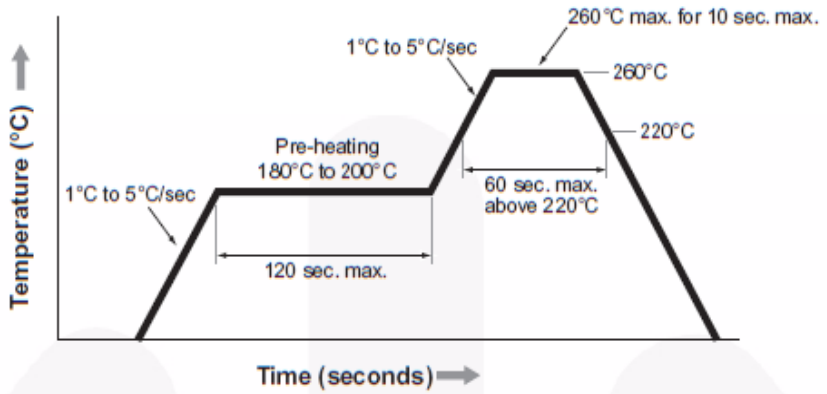
Dimensions in mm

Taping Dimensions for GR option



General tolerance ±0.1
Dimensions in mm

Reflow Profile



Note: Reflow soldering should not be done more than twice.

FAIRCHILD
SEMICONDUCTOR®

TRADEMARKS

The following includes registered and unregistered trademarks and service marks, owned by Fairchild Semiconductor and/or its global subsidiaries, and is not intended to be an exhaustive list of all such trademarks.

Auto-SPM™
Build it Now™
CorePLUS™
CorePOWER™
CROSSVOLT™
CTL™
Current Transfer Logic™
EcoSPARK®
EfficentMax™
EZSWITCH™*
E²CMOS™*
F®
Fairchild®
Fairchild Semiconductor®
FACT Quiet Series™
FACT®
FAST®
FastvCore™
FETBench™
FlashWriter®*
FPS™
F-PFS™
FRFET®
Global Power Resource℠
Green FPS™
Green FPS™ e-Series™
Gmax™
GTO™
IntelliMAX™
ISOPLANAR™
MegaBuck™
MICROCOUPLER™
MicroFET™
MicroPak™
MillerDrive™
MotionMax™
Motion-SPM™
OPTOLOGIC®
OPTOPLANAR®
®
PDP SPM™
Power-SPM™
PowerTrench®
PowerXS™
Programmable Active Droop™
QFET®
QS™
Quiet Series™
RapidConfigure™
™
Saving our world, 1mW/W/kW at a time™
SmartMax™
SMART START™
SPM®
STEALTH™
SuperFET™
SuperSOT™-3
SuperSOT™-6
SuperSOT™-8
SupreMOS™
SyncFET™
Sync-Lock™
SYSTEM ®*
GENERAL
The Power Franchise®
the power franchise
TinyBoost™
TinyBuck™
TinyLogic®
TINYOPTO™
TinyPower™
TinyPWM™
TinyWire™
TriFault Detect™
TRUECURRENT™*
μSerDes™
SerDes
UHC®
Ultra FRFET™
UniFET™
VCX™
VisualMax™
XS™

* Trademarks of System General Corporation, used under license by Fairchild Semiconductor.

DISCLAIMER

FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION, OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS. THESE SPECIFICATIONS DO NOT EXPAND THE TERMS OF FAIRCHILD'S WORLDWIDE TERMS AND CONDITIONS, SPECIFICALLY THE WARRANTY THEREIN, WHICH COVERS THESE PRODUCTS.

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION.

As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury of the user.
2. A critical component in any component of a life support, device, or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

ANTI-COUNTERFEITING POLICY

Fairchild Semiconductor Corporation's Anti-Counterfeiting Policy. Fairchild's Anti-Counterfeiting Policy is also stated on our external website, www.fairchildsemi.com, under Sales Support.

Counterfeiting of semiconductor parts is a growing problem in the industry. All manufacturers of semiconductor products are experiencing counterfeiting of their parts. Customers who inadvertently purchase counterfeit parts experience many problems such as loss of brand reputation, substandard performance, failed applications, and increased cost of production and manufacturing delays. Fairchild is taking strong measures to protect ourselves and our customers from the proliferation of counterfeit parts. Fairchild strongly encourages customers to purchase Fairchild parts either directly from Fairchild or from Authorized Fairchild Distributors who are listed by country on our web page cited above. Products customers buy either from Fairchild directly or from Authorized Fairchild Distributors are genuine parts, have full traceability, meet Fairchild's quality standards for handling and storage and provide access to Fairchild's full range of up-to-date technical and product information. Fairchild and our Authorized Distributors will stand behind all warranties and will appropriately address any warranty issues that may arise. Fairchild will not provide any warranty coverage or other assistance for parts bought from Unauthorized Sources. Fairchild is committed to combat this global problem and encourage our customers to do their part in stopping this practice by buying direct or from authorized distributors.

PRODUCT STATUS DEFINITIONS

Definition of Terms

Datasheet Identification	Product Status	Definition
Advance Information	Formative / In Design	Datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	Datasheet contains preliminary data; supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice to improve design.
No Identification Needed	Full Production	Datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice to improve the design.
Obsolete	Not In Production	Datasheet contains specifications on a product that is discontinued by Fairchild Semiconductor. The datasheet is for reference information only.

Rev. I40