



UNIVERSIDAD NACIONAL DE COLOMBIA

Control de una prótesis de mano para varios tipos de prensión empleando comandos de voz

Juan Pablo Ángel López

Universidad Nacional de Colombia

Facultad de Ingeniería

Bogotá, Colombia

2016

Control de una prótesis de mano para varios tipos de presión empleando comandos de voz

Juan Pablo Ángel López

Tesis presentada como requisito parcial para optar al título de:

Magister en Ingeniería Mecánica

Director:

Doctor Ingeniero Nelson Arzola de la Peña

Línea de Investigación:

Ingeniería de Diseño y Biomecánica

Grupos de Investigación:

Grupo de Investigación en Diseño Óptimo Multidisciplinario (UNAL)

Grupo de Investigación en Automática (Universidad Autónoma de Manizales)

Universidad Nacional de Colombia

Facultad de Ingeniería

Bogotá, Colombia

2016

“Las ciencias tienen las raíces amargas, pero muy dulces los frutos.”

Aristóteles

“El que no inventa, no vive.”

Ana María Matute

Agradecimientos

Gracias a Dios, a mi madre, a mi familia, a mi director, a mis amigos, a mis compañeros y a mis estudiantes. Gracias a la Universidad Nacional de Colombia por el aprendizaje y los conocimientos y gracias a la Universidad Autónoma de Manizales por las enseñanzas, el tiempo, la paciencia y la infraestructura, vitales para el desarrollo de esta investigación.

Resumen

Este documento presenta el desarrollo de un estudio cinemático de la mano empleando un sistema de captura de movimiento por cámaras infrarrojas y marcadores para posterior control de un prototipo de prótesis de mano. El objetivo es definir parámetros cinemáticos que permitan diferenciar cuatro tipos de agarre distintos y para ello se llevaron a cabo diferentes sesiones de captura de movimiento, logrando un total de 30 grabaciones por movimiento, logrando un total de 120 grabaciones. El análisis cinemático indica que existe una tendencia en las articulaciones de los dedos a presentar tres fases: acercamiento, cierre y sostenimiento. Centrando el estudio en la fase de cierre fue posible encontrar rangos angulares y duraciones de los cierres para los distintos tipos de agarre. Adicionalmente, se buscó desarrollar un sistema de control por movimientos (cierres o agarres) predefinidos en un prototipo de prótesis de mano, buscando cumplir con los parámetros cinemáticos hallados a partir de la captura de movimiento. Al final se logró construir un sistema de control en el entorno Matlab® que, enlazado con una tarjeta Arduino®, es capaz de reconocer diferentes comandos de voz y accionar el prototipo de prótesis para diferentes movimientos predefinidos.

Palabras clave: Miembros artificiales, electrónica, voz, biomecánica, lenguajes de programación, captura de movimiento.

Abstract

This document shows the development of a cinematic study of the hand using a motion capture system by infrared cameras and markers for subsequent control of a prototype prosthetic hand. The aim is to define kinematic parameters to differentiate four different types of grip, and for this purpose were held several motion capture sessions, achieving a total of 30 recordings by motion, making a total of 120 recordings. Kinematic analysis indicates a trend in the joints of the fingers to present three phases: approach, close and support. Focusing the study in the closing phase, it was possible to find angular ranges and duration of closures for different types of grip. Additionally, it was sought to develop a predefined motion control system (locks or grips) for the prototype prosthetic hand, looking to comply with the kinematic parameters found from the motion capture. Finally it was possible to build a control system in the Matlab® environment, linked to an Arduino® card, being able to recognize different voice commands and operate the prototype prosthesis for different predefined movements.

Keywords: Artificial Limbs, electronics, voice, biomechanics, programming languages, motion capture.

Contenido

	Pág.
Agradecimientos	VII
Resumen	IX
Contenido	XI
Lista de figuras	XIII
Lista de tablas	XV
Lista de Símbolos y abreviaturas	XVII
Introducción	XIX
1. Estado del Arte	3
1.1 Prótesis y Robots.....	3
1.2 Captura de Movimiento	6
1.3 Reconocimiento de Voz	8
2. Captura de Movimiento	13
2.1 Prótesis de Mano	13
2.2 Agarres y Posturas.....	16
2.3 Calibración del Sistema de Captura	17
2.4 Análisis Cinemático	21
2.4.1 Procesamiento de datos	21
2.4.2 Pre-procesamiento y variables angulares.....	22
2.4.3 Prueba piloto.....	23
2.4.4 Captura final.....	26
3. Control	31
3.1 Calibración de Sensores	31
3.2 Control en Lazo Abierto	37
3.3 Control de Movimientos por Agarres y Posturas Empleando Comandos de Voz.....	38
4. Resultados	49
4.1 Prueba Piloto.....	49
4.2 Captura Final	55
4.3 Control.....	59

5.	Conclusiones y Recomendaciones	65
5.1	Conclusiones	65
5.2	Recomendaciones	67
A.	Anexo: Adecuaciones al Prototipo de Prótesis Mano.....	69
B.	Anexo: Códigos escritos en el entorno MATLAB®	77
C.	Anexo: Manual de usuario para Interfaz de Control por Comandos de Voz.....	115
	Bibliografía	123

Lista de figuras

	Pág.
Figura 2-1. Prototipo de prótesis de mano [24].....	13
Figura 2-2. Agarres elegidos para ser simulados con la prótesis de mano. (a) Agarre cilíndrico. (b) Agarre esférico. (c) Agarre en pinza. (d) Agarre plano.	17
Figura 2-3. Posturas o posiciones estáticas a definir en la prótesis de mano. (a) Mano abierta. (b) "Bien". (c) Señalar.....	17
Figura 2-4. (a) Una de las ocho cámaras infrarrojas ubicadas en el laboratorio. (b) Haces infrarrojos emitidos por las cámaras.	18
Figura 2-5. (a) Marcadores de 3mm de diámetro. (b) Marcador para análisis de cuerpo entero. Fuente [26].....	19
Figura 2-6. Apariencia del software de captura de movimiento. Fuente [26].....	19
Figura 2-7. Ubicación de las cámaras para la prueba piloto.....	20
Figura 2-8. Esquema de composición de las matrices tridimensionales empleadas para el análisis.....	21
Figura 2-9. Ángulo calculado para la articulación MCF del dedo 2. (a) Antes de aplicar el proceso de filtrado. (b) Luego de aplicar el proceso de filtrado.	23
Figura 2-10. Representación de ubicación de marcadores y las rectas formadas por éstos para la definición de ángulos articulares.....	24
Figura 2-11. Visualización del dedo tras aplicar la transformación de coordenadas.	26
Figura 2-12. Ubicación de marcadores en las pruebas finales.	27
Figura 2-13. Visualización en perspectiva en Matlab® de los marcadores ubicados en la mano.	28
Figura 2-14. Distribución final de las cámaras. (a) Vista frontal. (b) Vista superior.	29
Figura 3-1. Interfaz de accionamiento para visualización de ángulos en la prótesis.....	32
Figura 3-2. Ubicación de marcadores en el Prototipo de Prótesis.	33

Figura 3-3. Ángulo calculado contra el valor entregado por el sensor para theta 1.	34
Figura 3-4. Ángulo calculado contra el valor entregado por el sensor para theta 2.	35
Figura 3-5. Ángulo calculado contra el valor entregado por el sensor para theta 3.	35
Figura 3-6. Ángulo calculado contra el valor entregado por el sensor para theta 4.	36
Figura 3-7. Ángulo calculado contra el valor entregado por el sensor para theta 5.	36
Figura 3-8. Interfaz gráfica para control en lazo abierto de la prótesis.....	38
Figura 3-9. Estructura jerárquica de un sistema basado en ANS [38].	39
Figura 3-10. Capas de una Red Neuronal Artificial [38].....	40
Figura 3-11. Arquitectura red neuronal para control en lazo cerrado.	42
Figura 3-12. Esquema de funcionamiento de la red neuronal que permite el reconocimiento de los comandos de voz.....	43
Figura 3-13. Funciones de activación. La elegida es la función sigmoidea [38].	45
Figura 3-14. Apariencia de la interfaz gráfica para control por voz.....	47
Figura 4-1. Representación gráfica de las fases de acercamiento (1), cierre (2) y sostenimiento (3) presentadas en la mayoría de las articulaciones de la mano.....	49
Figura 4-2. Marcadores posicionados en el dorso de la mano.	56
Figura 4-3. Ilustración de variables extraídas por cada articulación.	56
Figura 4-4. Valores de los <i>sliders</i> , valores de los sensores y posición de la mano de acuerdo a éstos.....	59
Figura 4-5. Funcionamiento del botón "Entrenar".	60
Figura 4-6. Funcionamiento del botón "Escuchar".	61
Figura 4-7. Prototipo de prótesis realizando la acción "Bien".	62

Lista de tablas

	Pág.
Tabla 2-1. Dimensiones en milímetros (mm) del prototipo de prótesis [23].	14
Tabla 2-2. Dimensiones de la palma de la mano (mm) [23].	14
Tabla 2-3. Ángulos de flexión de los dedos 2 a 5 [23].....	15
Tabla 2-4. Medidas de la mano derecha tomadas en los individuos que participaron en la captura de movimiento.....	27
Tabla 4-1. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo anular en agarre cilíndrico.....	50
Tabla 4-2. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo corazón en agarre cilíndrico.	50
Tabla 4-3. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo índice en agarre cilíndrico.	50
Tabla 4-4. Ángulos máximos y mínimos y definición de la pendiente de cierre en las cuatro articulaciones del dedo pulgar en agarre cilíndrico.....	50
Tabla 4-5. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo anular en agarre esférico.....	51
Tabla 4-6. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo corazón en agarre esférico.	51
Tabla 4-7. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo índice en agarre esférico.	51
Tabla 4-8. Ángulos máximos y mínimos y definición de la pendiente de cierre en las cuatro articulaciones del dedo pulgar en agarre esférico.....	51
Tabla 4-9. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo anular en agarre plano.	51

Tabla 4-10. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo corazón en agarre plano.....	52
Tabla 4-11. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo índice en agarre plano.....	52
Tabla 4-12. Ángulos máximos y mínimos y definición de la pendiente de cierre en las cuatro articulaciones del dedo pulgar en agarre plano.	52
Tabla 4-13. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo índice en agarre en pinza.	52
Tabla 4-14. Ángulos máximos y mínimos y definición de la pendiente de cierre en las cuatro articulaciones del dedo pulgar en agarre en pinza.....	52
Tabla 4-15. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo anular en mano abierta.....	53
Tabla 4-16. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo corazón en mano abierta.	53
Tabla 4-17. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo índice en mano abierta.....	53
Tabla 4-18. Ángulos máximos y mínimos y definición de la pendiente de cierre en las cuatro articulaciones del dedo pulgar en mano abierta.....	53
Tabla 4-19. Datos arrojados para el agarre Cilíndrico.	57
Tabla 4-20. Datos arrojados para el agarre Esférico.....	57
Tabla 4-21. Datos arrojados para el agarre Plano.	57
Tabla 4-22. Datos arrojados para el agarre en Pinza.	57
Tabla 4-23. Ejemplos de cálculo del error relativo de la RNA.....	60
Tabla 4-24. Porcentaje de éxito del criterio de parada.	62

Lista de Símbolos y abreviaturas

Símbolos con letras latinas

Símbolo	Término	Unidad SI	Definición
A	Área	m ²	$\iint dx dy$
a	Coeficiente	1	
D	Diámetro	m	
L	Longitud	m	
m	Pendiente	m	$\frac{y_2 - y_1}{x_2 - x_1}$
t	Tiempo	s	
V	Volumen	m ³	$\int dr^3$
\vec{u}	Velocidad	$\frac{m}{s}$	$\frac{d\vec{r}}{dt}$
f_i	Flexión	bits	

Símbolos con letras griegas

Símbolo	Término	Unidad SI	Definición
α	Velocidad angular	$\frac{rad}{s}$	$\frac{d\theta}{dt}$
θ	Ángulo interno entre vectores	rad	
Δ	Rango de cambio	1	$valor_{final} - valor_{inicial}$

Subíndices

Subíndice	Término
0	Estado de referencia

Superíndices

Superíndice	Término
n	Exponente, potencia

Abreviaturas

Abreviatura	Término
DOF	Grados de libertad
RNA	Red Neuronal Artificial
MoCap	Captura de movimiento
IF	Articulación Interfalángica
MCF	Articulación metcarpofalángica
OP	Articulación de oposición del pulgar

Introducción

El análisis de movimiento en el cuerpo humano es de gran importancia para diferentes disciplinas, entre ellas la medicina, fisioterapia y rehabilitación, ciencias deportivas e ingeniería. Ésta última, sin ser menos importante, es la encargada de describir y explicar, por medio de modelos matemáticos, el comportamiento de las estructuras biológicas que intervienen en ciertos movimientos de interés para alguna investigación en particular.

Desde el punto de vista de la ingeniería, el análisis del movimiento corporal permite estudiar propiedades biológicas y de materiales y diferentes características mecánicas con el fin de entender el comportamiento del cuerpo humano [1]. Estas descripciones han facilitado el diseño de prótesis y órtesis, el diseño de dispositivos para tratamientos circulatorios, mejoras en los procedimientos quirúrgicos, entre otros.

Particularmente, para el diseño de prótesis y órtesis han sido de gran utilidad las técnicas de análisis de movimiento por medio de cámaras de video y laboratorios de biomecánica. Estos laboratorios brindan información cinética y cinemática de las extremidades para luego formular modelos matemáticos que representen desplazamientos, velocidades, aceleraciones, fuerzas y momentos [2]. Las descripciones tomadas a partir de cámaras de video, en un principio, brindaban información únicamente bidimensional y posteriormente se superponía la información de hasta tres cámaras ubicadas en diferentes planos para construir modelos en tres dimensiones. Este tipo de análisis se realizaba por medio de fotogramas, extrayendo cuadro por cuadro en una secuencia de video los datos de interés para el modelo matemático en construcción.

Sin embargo, actualmente existen diversos métodos de captura y análisis de movimiento integrados que arrojan coordenadas espaciales de las articulaciones o de los segmentos

corporales. Para este tipo de análisis generalmente se hace uso de marcadores que son detectados por el sistema de captura, los cuales son ubicados en lugares específicos del cuerpo humano para obtener la información requerida [2].

En el ámbito de las prótesis de mano robotizadas, existen diferentes tipos de dispositivos con variadas estrategias de control, entre las cuales el control mioeléctrico es el más común. Este tipo de accionamiento consiste en ubicar electrodos de superficie sobre la piel y a través de estos capturar señales eléctricas producidas por los músculos; utilizando dichas señales como entradas a un sistema de control donde la salida es el movimiento del dispositivo protésico.

En usuarios de prótesis de mano, las señales mioeléctricas manipuladas provienen de músculos residuales o adyacentes a la amputación, requiriendo esto que la persona acuda a tratamientos y terapias que le permitan contraer dichos músculos a voluntad, normalmente en formas que no son naturales. Este acondicionamiento del paciente puede tomar semanas y hasta meses para lograr un desempeño óptimo, teniendo en cuenta que el resultado es lograr movimientos que no son naturales y algunas veces incómodos.

Es a partir de lo anterior que surge la pregunta:

¿Es posible desarrollar una estrategia de control empleando señales de voz, que permita obtener posturas y movimientos bioinspirados, en una prótesis de mano robótica de 7 grados de libertad?

Este proyecto de investigación pretende desarrollar una estrategia de control por computador que permita simular ciertos tipos de presión propios de la mano humana, la cual posee más de 20 grados de libertad, con una prótesis robótica que únicamente posee siete grados de libertad.

En este tipo de dispositivos se busca tomar decisiones de control en tiempo real, lo que implica diseñar algoritmos robustos que procesen información de manera eficiente y rápida, para que finalmente pueda existir una realimentación visual inmediata para quien esté usando el elemento protésico. Otra de las condiciones de control en prótesis está ligada a la ergonomía y

facilidad de entrenamiento y uso del mecanismo. Es decir, hay que ser cuidadosos con la cantidad y la forma como se toman las señales de control, de tal manera que no se sacrifique la comodidad del usuario a la vez que para éste sea fácil el control del elemento biónico.

Para el caso particular de la prótesis objeto de esta investigación, el interés se enfoca hacia el diseño de una rutina que, por medio de diferentes comandos de voz, logre controlar dicho dispositivo para que a partir de ciertas posturas predeterminadas, cumpla aproximadamente con algunas de las diferentes funciones que puede brindar una mano humana. Así, los objetivos planteados para esta la investigación son:

Objetivo General

Diseñar e implementar una estrategia de control que permita generar tipos de prensión bioinspirados en una prótesis de mano de siete grados de libertad.

Objetivos Específicos

- Analizar geométrica y cinemáticamente en una mano real, cada tipo de prensión a simular mediante procesamiento digital de imágenes.
- Describir matemáticamente el modelo cinemático asociado a cada tipo de prensión a simular.
- Transcribir las propiedades geométricas y cinemáticas obtenidas para una mano al modelo cinemático de la prótesis.
- Desarrollar rutinas de control mediante comandos de voz para la prótesis.
- Evaluar resultados de la cinemática en un prototipo de prótesis de siete grados de libertad (Promanu).

La información cinemática de la mano se obtuvo mediante un sistema de captura de movimiento compuesto por ocho cámaras infrarrojas, cada una con su propio procesamiento digital de imágenes; capaz de entrecruzar la información entregada por cada una de las cámaras y así construir con una alta precisión matrices de coordenadas 3D para una nube de marcadores [3]. El sistema mencionado se encuentra en la ciudad de Manizales, en las instalaciones de la Universidad Autónoma de Manizales (UAM®), en el Laboratorio de Biomecánica.

Tras obtener la información cinemática de la mano, se lograron definir posturas y agarres a ser programados en un prototipo de prótesis de mano de siete grados de libertad, el cual se controla por medio de comandos de voz, detectados por una Red Neuronal Artificial, a través de una interfaz gráfica construida en el entorno Matlab®.

Este documento inicia con una descripción del Estado del Arte relacionado con la investigación, dividido en tres partes: Prótesis y robots, captura de movimiento y reconocimiento de voz; destacando en cada área los antecedentes más relevantes. En la segunda parte del documento se explica en detalle la metodología desarrollada para llevar a cabo las pruebas de captura de movimiento, los agarres a parametrizar y los equipos y dispositivos empleados para tal fin, iniciando con una prueba piloto para calibración del sistema y definición de parámetros para el protocolo experimental de las pruebas finales. En el tercer apartado se muestra la implementación del sistema de control por diferentes etapas, partiendo desde un simple control en lazo abierto para manipulación del prototipo, llegando hasta un control en lazo cerrado por comandos de voz. Al final del informe se consignan los resultados de las pruebas de captura de movimiento y de la etapa de control, seguido por las conclusiones y las recomendaciones.

1. Estado del Arte

En este apartado se mencionan los antecedentes más relevantes para la presente investigación. Para el desarrollo del proyecto se diferenciaron tres aspectos o etapas principales a abordar, inicialmente de forma independiente, para luego en conjunto conformar el producto final de este trabajo.

1.1 Prótesis y robots

En su tesis, Alonso [4] detalla la obtención de los modelos cinemático y dinámico de una mano robótica antropomórfica de cuatro dedos y cuatro grados de libertad en cada uno. Los dedos de la mano se encuentran completamente actuados por medio de bandas conectadas a motores DC. Este trabajo de grado pretende demostrar que la ley de control basada en pasividad tiene un mejor comportamiento en lo que respecta al error, comparada con el control calculado. Como resultados se pudo observar que ninguna de las articulaciones de los dedos excede unos límites de movimientos determinados; también se hace evidente, por medio de simulación en MATLAB® que empleando un método de control calculado existen oscilaciones indeseadas que pueden ser disminuidas empleando el método de control pasivo; al comparar el desempeño de los dos tipos de control mencionados, se demuestra que por medio del control pasivo se reduce el error a la vez que las acciones de control se ejecutan en un tiempo menor. Siguiendo la línea de este proyecto sería necesario ampliar las posibilidades de este dispositivo agregando un dedo adicional y así obtener un diseño más antropomórfico. Vale destacar los avances que esta investigación aporta a las estrategias de control de prótesis pues se logró un control óptimo de 4 dedos con 4 grados de libertad cada uno, es decir, un total de 16 grados de libertad.

Otra tesis de grado, desarrollada por Pilaquina [5], está orientada hacia el desarrollo de una mano robótica controlada mediante un guante. El objetivo del dispositivo fue el de seguir con precisión los movimientos de la mano inmersa en el guante de control. En el proceso de

investigación se logró determinar que el error total asociado a la posición de la mano robótica fue de 1,55% al controlarla con el guante y de 0,46% al ser controlada por parámetros angulares en cada articulación por medio de una interfaz gráfica de computador. Esta investigación muestra una técnica que puede ser de gran utilidad al momento de describir mecánicamente y geoméricamente los movimientos de la mano y motiva a mejorar en la precisión de movimientos simulados con manos mecatrónicas.

Yang y otros [6] presentaron una mano con múltiples dedos actuados; cuatro con dos articulaciones cada uno y el pulgar con tres articulaciones. Cada articulación fue diseñada con un mecanismo flexible basada en la carga de un resorte de compresión en las direcciones axial y transversal usando un sistema de guayas. Se obtuvo como resultado una mano de cinco dedos actuados, cada uno con tres grados de libertad, empleando un sistema de carga de resortes de compresión en los sentidos axial y transversal que permite una manipulación controlable manteniendo el dominio en cada dedo. El mecanismo de guayas permite ubicar el sistema de accionamiento lejos de la mano. Este proyecto abre un campo de posibilidades asociadas a la optimización e innovación en diseños mecánicos que pretendan imitar los movimientos y funciones de la mano humana.

Pérez y Mendoza [7] desarrollaron una investigación que buscó el desarrollo de una mano robótica que pudiese imitar los movimientos de una mano humana tocando piezas musicales en un piano digital. Para el proyecto se aplicaron las metodologías y herramientas para el diseño de sistemas digitales, diseños de circuitos y sistemas electrónicos. Al final se obtuvo como resultado un dispositivo compuesto por una mano de tres dedos y un sistema de riel que le permitiera a la mano ubicarse frente a las teclas que debían ser oprimidas, y ser capaz de reproducir diferentes melodías programadas por medio de notas musicales. Sería necesario continuar esta investigación agregando el tipo de precisión lograda con este mecanismo a un diseño completo de mano de cinco dedos, con diseño antropomórfico para posteriores aplicaciones en prótesis.

El dispositivo MA-I (Mano Artificial Inteligente), desarrollado por Suárez y Grosch [8], es un manipulador robótico con aplicación como efector final en un robot industrial, de características

antropomórficas, es decir, cumple con parámetros de dimensión y ubicación de dedos y falanges propios de una mano real con la diferencia que el dispositivo mencionado posee únicamente 4 de los 5 dedos que componen una mano humana. Este sistema empleó un control independiente para cada una de las articulaciones, lo que se tradujo en un total de 16 lazos de control independientes; también se utilizaron sensores en las yemas de los dedos. El resultado de este proyecto fue la construcción de una mano robótica modular, de yemas de dedos intercambiables, y versátil a la hora de aplicar distintas estrategias de control.

Por su parte, Sarmiento y otros [9] afirman que el proceso investigativo expuesto en su artículo realiza un aporte en la búsqueda de alternativas tecnológicas que mejoren la autonomía de los usuarios de elementos protésicos, teniendo en cuenta un metodología apropiada en la adaptación del nuevo miembro por medio del aprendizaje de rutinas motoras. El proyecto tiene como objetivo evaluar la interacción de una prótesis mecatrónica, controlada por señales mioeléctricas o por señales de voz emitidas por una persona con amputación traumática a nivel del tercio distal del codo, que le permita aprender a realizar de forma más eficiente y eficaz la actividad motora de procesos prensiles de precisión. En la investigación se evaluó la eficiencia y eficacia en el aprendizaje de actividades motoras de prensión, mediante una prótesis mecatrónica como ayuda aumentativa para discapacitados con amputación entre el codo y la muñeca. Se implementaron dos estrategias independientes de control; una con señales mioeléctricas y otra con comandos de voz. Las señales mioeléctricas se midieron como pulsos eléctricos, cada uno tomado en cuenta si éste superaba un umbral definido; estos pulsos fueron contados y según ese conteo se definía una acción particular. Para los comandos de voz se definieron cinco palabras diferentes, cada una asociada a una tarea particular de la prótesis. Como resultado de la investigación se pudo concluir que en una visión global hubo mayor eficiencia y eficacia en la rutina mediante comandos de voz aunque en ciertas acciones presentaron una mejor respuesta con señales mioeléctricas; también se pudo inferir que el proceso cognitivo del habla permite una mayor consolidación de la acción motriz de prensión que un impulso mioeléctrico.

En el trabajo de investigación presentado por Asyali y otros [10] se muestran las ventajas que, según dichos autores, presenta el control mediante comandos de voz sobre el control a partir

de señales mioeléctricas. Se diseñó y construyó una prótesis de mano de tres grados de libertad con las funciones de apertura y cierre para agarre de distintos objetos. En el proceso se implementó un control electrónico mediante comandos de voz usando palabras como “*pick up*” (recoger) y “*release*” (soltar). Los autores argumentan que la voz permite una mayor gama de acciones y confort, que las señales mioeléctricas no tienen a la vez que no requiere un entrenamiento muy extenso previo al uso de la prótesis.

1.2 Captura de movimiento

El artículo presentado por Feng y otros [11] propone un enfoque diferente a los convencionales para extraer características de contorno y forma en imágenes de la mano. Dicho enfoque se compone por dos pasos, la fase de ubicación gruesa (CLP en inglés) y la fase de ubicación detallada (RLP) desde la rudeza hacia el refinamiento. En la fase CLP, el contorno de la mano se describe de manera aproximada, por un polígono con concavidad y convexidad; se discute meticulosamente un enfoque para obtener el polígono de forma de la mano usando puntos y líneas de ubicación. Después se propone un algoritmo de ubicación gruesa (CL en inglés) para extraer características de interés en la forma de la mano como contorno, yemas de los dedos, raíces de los dedos, articulaciones e intersecciones de los nudillos en los diferentes dedos. En la fase RLP se introduce un enfoque multiescala para detallar las características obtenidas por el algoritmo CL. También se propone un algoritmo de ubicación refinada (RL) para efectos de definir la fuerza de respuesta en diferentes tipos de características. El aporte que se expone en el documento es el método novedoso que permite obtener por medio de los dos pasos anteriores modelos 2D y 3D de la mano sin necesidad de usar ningún tipo de marcadores o implementos extraños sobre la mano, lo cual permite independizar el procesamiento digital de imágenes de los implementos utilizados para registrar los movimientos y posturas de la mano.

En el documento presentado por Yin y Xie [12] se muestra un sistema de reconocimiento de gestos de la mano implementado en un robot humanoide de servicio. El sistema aplica redes

neuronales RCE¹ basadas en un algoritmo de segmentación de colores para separar las imágenes de la mano de fondos complejos. Las características topológicas de la mano son extraídas de la silueta de la región segmentada de la mano. Se logran identificar con precisión posturas de la mano, basado en el análisis de características simples pero distintivas. La red neuronal es capaz de caracterizar la distribución de la región de la mano para todos los colores de piel; el reconocimiento de las posturas de la mano está basado en características topológicas de la mano que se extraen de la imagen procesada. El sistema final cuenta con las siguientes características: robustez y dinámica para fondos complejos, adaptabilidad para variaciones de luminosidad, invariancia rotacional, desempeño en tiempo real e independencia entre el usuario y el dispositivo. Se lograron detectar ocho posturas de la mano para la programación del robot anteriormente mencionado y los resultados experimentales demostraron la efectividad y robustez del sistema.

El proceso de investigación llevado a cabo por Wu y Huang [13] se ve motivado por la búsqueda de nuevos dispositivos que permitan una interacción más cómoda entre las computadoras y los seres humanos. La estrategia presentada por los investigadores para lograr esta interacción consiste en reconocer diferentes tipos de posturas de la mano para luego asociar cada postura a una acción particular. Mediante un procesamiento de imágenes con dos enfoques, uno global y otro local, realizan un análisis dinámico de la mano en diferentes posturas para así determinar las características asociadas a cada configuración. Localmente analizan cada uno de los dedos como mecanismos independientes determinando las velocidades y posiciones de las falanges; globalmente determinan por medio de un análisis cromatológico la geometría de la palma y la posición de la mano. Al final se logró obtener un modelamiento en la forma, la estructura y el comportamiento dinámico para describir el movimiento de la mano y así poder identificar las diferentes características asociadas a una postura en particular.

¹ Reduced Coulumb Energy por sus siglas en inglés.

1.3 Reconocimiento de Voz

El sistema de reconocimiento de comandos de voz es una aplicación del procesamiento digital de señales, que permite que el usuario interactúe y grabe una serie de palabras utilizando un micrófono. El software procesa los datos convirtiéndolos en comandos y diferenciándolos mediante el espectro de frecuencias de la señal de voz, que oscilan entre los 500 Hz y 3500 Hz [14] e incluyen los tonos fundamentales de la voz con sus respectivos armónicos. Estos datos se obtienen mediante la aplicación de la transformada rápida de Fourier (FFT) ya que “la representación de la frecuencia –eje horizontal– y la amplitud –eje vertical– de los armónicos en un instante de la señal sonora del habla” [15].

Una de las técnicas más empleadas en este campo según [16] es la aplicación de redes neuronales, las cuales corresponden a diversas unidades de procesamiento (que se denominan neuronas) y se entrenan utilizando los datos de entrada y salida de la red para luego producir resultados satisfactorios al momento de probarse con datos similares; es decir, reconoce comandos. El objetivo del estudio consiste en reconocer diez números (en el idioma turco) mediante procesamiento digital de señales, que consiste en: enmarcar, ventaneo, transformada rápida de Fourier y conversión de coeficientes de frecuencia *cepstral* (representa el espectro de potencia en pequeños periodos de tiempo [17]). Los resultados obtenidos por este método mostraron tasas de reconocimiento entre el 98% y 99,5%, lo que indica que el uso de redes neuronales artificiales es eficaz y práctico para el reconocimiento de comandos de voz.

El desarrollo de una interfaz de voz inteligente que permita que un carro entienda los comandos de voz humanos fue el trabajo de los autores [18]. Se hizo posible combinando un sistema de reconocimiento de voz que compara coeficientes *cepstrum* de las frecuencias de voz con las emisiones infrarrojas emitidas por un modelo particular de vehículo a través de un micro chip. Dicha combinación podría alcanzar hasta un 80% de precisión. Teniendo en cuenta que al momento de probar el modelo son muchos los factores que pueden interferir y alterar los resultados, se adaptó el modelo para reconocer el coeficiente *cepstrum* más cercano al obtenido tanto para una sola palabra con el 90% de eficiencia como para dos palabras con un 80%, luego de diversas pruebas y experimentos.

De igual forma, en el desarrollo de interfaces interactivas con Matlab®, se tiene que [19], a través del *toolbox* SimuLink®, crearon un simulador donde los comandos de voz pueden ser manipulados a través de una serie de opciones que el usuario puede elegir. Esto se obtiene mediante la implementación del método de análisis predictivo lineal, el cual convierte una señal en un modelo matemático, donde se calculan una serie de coeficientes (LPC) partiendo de ésta. Luego, la señal se filtra inversamente y el resultado de este filtrado se conoce como señal residual que se considera como una estimación de la excitación del habla; además que al integrarla puede obtenerse el volumen de la voz. Durante la investigación se hizo evidente que existe una serie de factores que podrían afectar el estudio. Para dar solución a esto se consideró que al utilizar una voz sintética se reducirían a cero las variaciones ambientales y por el contrario podrían cambiarse diversos factores en cuanto a la voz y examinar los efectos que produce.

Romero y Álvarez [20] llevaron a cabo un proyecto que fue presentado como resultado de la investigación en métodos para mejorar la calidad de vida de personas discapacitadas en el marco de ejecución de acciones típicas de la cotidianidad. El prototipo esencialmente está formado por cuatro tarjetas y sensores de final de carrera que le permiten interactuar con sus cargas según las acciones de control emitidas por el usuario. El funcionamiento del sistema se describe como una interacción de diferentes módulos, partiendo desde una conversión análogo-digital de comandos de voz; posteriormente se emplean técnicas espectrales, filtrados y distintos tipos de procesamiento digital de señales; luego se cuantifican los sonidos a partir de vectores generados en el DSP; como paso siguiente se reconoce una secuencia de voz como datos conocidos o desconocidos que puede traducirse en una pronunciación asociada a una acción de control. Tras aplicar el sistema de reconocimiento de voz se logró controlar un grupo de distintos dispositivos asociados a acciones cotidianas como encender y apagar luces y ventiladores, controlar televisores o abrir y cerrar puertas o cortinas. Este proyecto motiva a generar más desarrollos tecnológicos para permitir que las personas en situación de discapacidad tengan la oportunidad de interactuar más activamente con los objetos y las personas que los rodean.

En el artículo presentado por Sakoe y otros [21] se estudia el comportamiento de un algoritmo cuyo funcionamiento consiste en realizar una programación dinámica por pareo de patrones con un efecto no lineal de normalización en el tiempo para el reconocimiento de la voz. Durante el proceso se propuso comparar dos tipos de patrones (asimétrico y simétrico) con cuatro pruebas algorítmicas diferentes en donde cada una era ejecutada por una persona diferente pronunciando distintos tipos de palabras; los resultados fueron comparados con datos teóricos evaluando el porcentaje de error promedio. Tras confrontar los algoritmos de programación dinámica con otros algoritmos con funciones similares se pudo observar un comportamiento superior en las rutinas expuestas en el artículo. También se hizo evidente que los patrones simétricos tenían mejor comportamiento que los asimétricos.

En el proyecto de investigación desarrollado por Halavati y otros [22] se realizó un análisis espectral de la voz por medio de imágenes y sus colores respectivos para diferentes fonemas. Para tener un sistema de reconocimiento de voz rápido y robusto, este documento presenta un modelo difuso de señal de voz mediante la representación de espectrogramas de voz con términos lingüísticos. La principal diferencia con los enfoques no difusos anteriores a esta investigación consiste en ignorar ciertos detalles que hacen que el proceso de reconocimiento sea lento y sensible a pequeñas perturbaciones o ruidos. La principal diferencia con otros modelos difusos existentes está en el hecho de que este modelo hace un uso no convencional de las características acústicas de la señal de voz y se basa totalmente en el pensamiento difuso, utilizando las nuevas características de la señal de voz que se definen por términos lingüísticos gruesos, muy parecido a la forma como un ser humano lee y traduce un espectrograma de habla en las clases de fonemas. El enfoque se pone a prueba sobre una base de datos estándar de un único hablante y el reconocimiento de múltiples hablantes tanto para ambientes ruidosos como ambientes libres de ruido. El algoritmo ha demostrado ser bastante exitoso en el mapeo de las muestras de los fonemas y ha logrado ignorar el ruido en lugar de su detección y eliminación. Mientras que el método propuesto es mucho más simple y utiliza menos energía computacional, ha ganado importantes y mejores resultados en el manejo del ruido y el reconocimiento en ambientes muy ruidosos.

2. Captura de Movimiento

2.1 Prótesis de Mano

Esta investigación planteó como propósito principal adecuar, instrumentar y controlar un prototipo de prótesis de mano (Figura 2-1) desarrollado en la Universidad Nacional de Colombia, Sede Bogotá, como resultado de investigación de una tesis de Maestría en Ingeniería Mecánica. Este dispositivo fue fabricado por medio de la técnica de prototipado rápido por fotocurado de resina polimérica [23]. A continuación se hará una breve descripción de las características mecánicas de la prótesis, se mencionarán algunas adecuaciones y modificaciones con respecto al diseño original y posteriormente se explicará la instrumentación electrónica del mecanismo.



Figura 2-1. Prototipo de prótesis de mano [24].

El prototipo de prótesis que sirvió como insumo principal para esta investigación cuenta con siete grados de libertad, distribuidos de la siguiente manera:

1. Oposición del dedo 1 (pulgar).
2. Flexo-extensión del dedo 1.
3. Flexo-extensión del dedo 2 (índice).
4. Flexo-extensión del dedo 3 (corazón).
5. Flexo-extensión de los dedos 4 y 5 (anular y meñique), que se mueven en conjunto.
6. Flexo-extensión de la muñeca.
7. Rotación del antebrazo (visto como prono-supinación) [23] [24].

Las dimensiones y parámetros angulares del prototipo de prótesis de mano se consignan a continuación, desde la Tabla 2-1 hasta la Tabla 2-3.

Tabla 2-1. Dimensiones en milímetros (mm) del prototipo de prótesis [23].

Falange Dedos	Largo				Ancho			Espesor		
	Falange Proximal	Falange Medial	Falange Distal	Total	Unión con la palma	En la última falange	Promedio	Unión con la palma	En la última falange	Promedio
Dedo 1	37	-	31	68 (34)	19	15	17	20	11	15,5
Dedo 2	32	26	18	76 (33)	16 (28)	13 (29)	14,5	13	11	12
Dedo 3	35	29	20	84 (32)	16 (26)	13 (27)	14,5	13	11	12
Dedo 4	30	28	28	78 (31)	16 (24)	13 (25)	14,5	12	10	11
Dedo 5	25	20	17	62 (30)	14 (22)	11 (23)	12,5	11	9	10

*Los números en paréntesis corresponden a las mediciones representadas en la Figura 2-2.

Tabla 2-2. Dimensiones de la palma de la mano (mm) [23].

Dimensión	Valor
Largo	110 (35)
Ancho con pulgar	90
Ancho sin pulgar	75
Espesor lado pulgar	25
Espesor lado opuesto al pulgar	16

Tabla 2-3. Ángulos de flexión de los dedos 2 a 5 [23].

Articulación	Rango	Observaciones
IFD	90°	Los ángulos de flexión para los dedos, se tomaron iguales para los dedos, índice, corazón, anular y meñique. Se tomaron también respecto a una horizontal coincidente con el eje axial del dedo
IFP	110°-135°	
MCF	90°-110°	

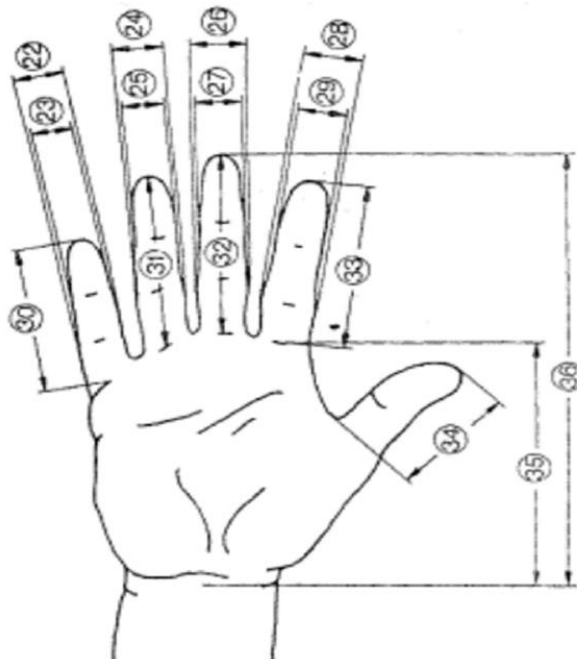


Figura 2-2. Dimensiones lineales de la mano [23].

Los ángulos de aducción-abducción entre los dedos (entre 2 y 3, entre 3 y 4, entre 4 y 5) son de 7° en los tres casos. En un plano paralelo a la palma de la mano, el ángulo entre los dedos 1 y 2 es de 90°. Las articulaciones del pulgar (MCF, IF y OP) presentan rangos angulares de 0° a 90° [23].

Los dedos 2 a 5 cuentan con las articulaciones metacarpo-falángica (MCF), interfalángica proximal (IFP) e interfalángica distal (IFD), las cuales se accionan simultáneamente en cada dedo. El dedo 1 presenta una articulación de oposición (OP) independiente y las articulaciones interfalángica (IF) y MCF que actúan simultáneamente. Los cierres (flexión) de cada uno de los dedos son generados por guayas (cuerdas de nylon) conducidas al interior de éstos, que se conectan a poleas [23] [24].

Con el fin de generar en la prótesis movimientos similares a los generados por la mano humana, se decidió partir del análisis cinemático derivado de diferentes pruebas de captura de movimiento, tomando como base ciertas posturas y agarres definidos (Figura 2-3). Para esto se contó con un Laboratorio de Biomecánica equipado con doce cámaras infrarrojas que detectan marcadores ubicados en el cuerpo. Dado que este sistema fue fabricado para capturar movimientos de cuerpo entero (movimientos espaciales con marcadores de gran tamaño) o movimientos faciales (acciones en un solo plano con marcadores de menor tamaño), se identificó la necesidad de diseñar una ubicación de cámaras no estandarizada por el fabricante, con el objetivo de capturar movimientos espaciales con marcadores de menor tamaño. Así pues, este apartado muestra y analiza información obtenida en las pruebas de análisis cinemático de la mano para varios tipos de prensión y posturas utilizando un sistema de captura de movimiento.

En una primera instancia se describe el proceso de adecuación y calibración del sistema de captura de movimiento, el cual, por estar diseñado para captura de cuerpo entero tuvo que ser modificado para análisis de la mano. Posteriormente, se describe el proceso de análisis de datos obtenidos en las pruebas de análisis de agarres y posturas para construir los modelos matemáticos a transferir a la prótesis de mano. En esta etapa se llevó a cabo un análisis por cada tipo de agarre y postura para cada dedo independientemente. Se lograron calcular y visualizar el cambio de los ángulos con respecto al tiempo en las articulaciones IF, IFP, IFD y MCF en los dedos pulgar, índice, corazón y anular.

2.2 Agarres y Posturas

Se eligieron cuatro agarres (Figura 2-3) y tres posturas (Figura 2-4) para ser simulados con la prótesis de mano. Se definieron dos agarres “gruesos” (cilíndrico y esférico), dos agarres “finos” (plano y pinza) y tres posturas o posiciones estáticas (mano abierta, señalar y la señal “bien”). Estos agarres y posturas fueron escogidos por ser considerados viables para ser generados con el dispositivo protésico y por ser comúnmente ejecutados en la cotidianidad.

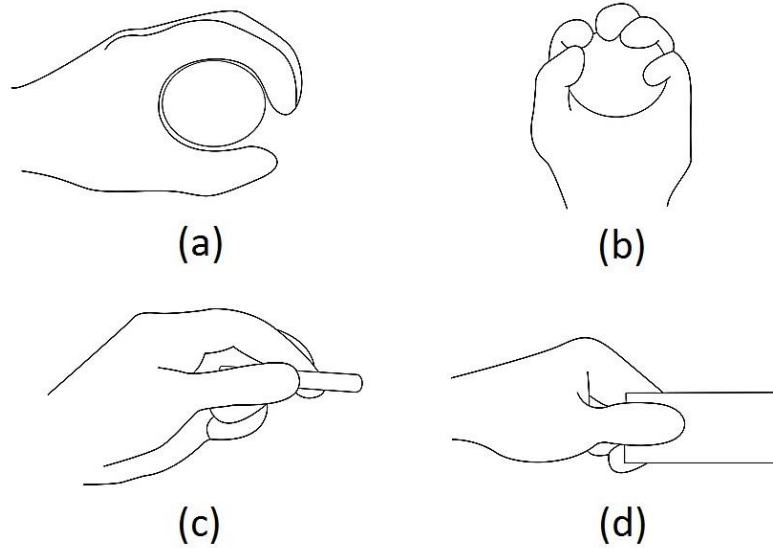


Figura 2-3. Agarres elegidos para ser simulados con la prótesis de mano. (a) Agarre cilíndrico. (b) Agarre esférico. (c) Agarre en pinza. (d) Agarre plano.

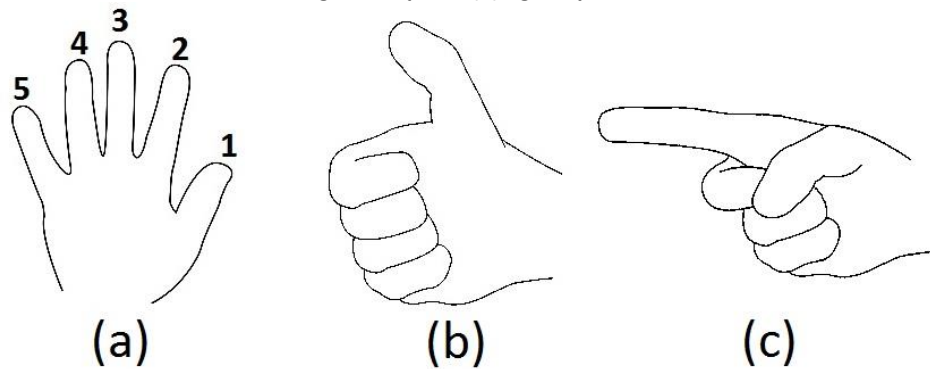


Figura 2-4. Posturas o posiciones estáticas a definir en la prótesis de mano. (a) Mano abierta. (b) "Bien". (c) Señalar.

2.3 Calibración del Sistema de Captura

El sistema de captura de movimiento con el cual fueron realizadas las pruebas está compuesto por un conjunto de doce cámaras infrarrojas (de las cuales se utilizaron ocho); dos concentradores, cada uno para conectar hasta seis cámaras; un software de visualización y exportación de datos; y diferentes elementos de infraestructura como bases, trípodes y cables USB.

Cámaras

Las cámaras empleadas en el Laboratorio de Biomecánica de la UAM® (Figura 2-5) cuentan cada una con su propio procesamiento digital de imágenes en escala de grises, en donde se destaca y se envía al software de visualización y exportación la coordenada de los píxeles que muestren reflexión de luz infrarroja. Cada cámara posee 26 LEDs infrarrojos que envían haces de luz hacia el área de captura y que son reflejados por ciertos marcadores; es esta reflexión lo que detectan las cámaras mencionadas para construir las matrices de coordenadas 3D [3].

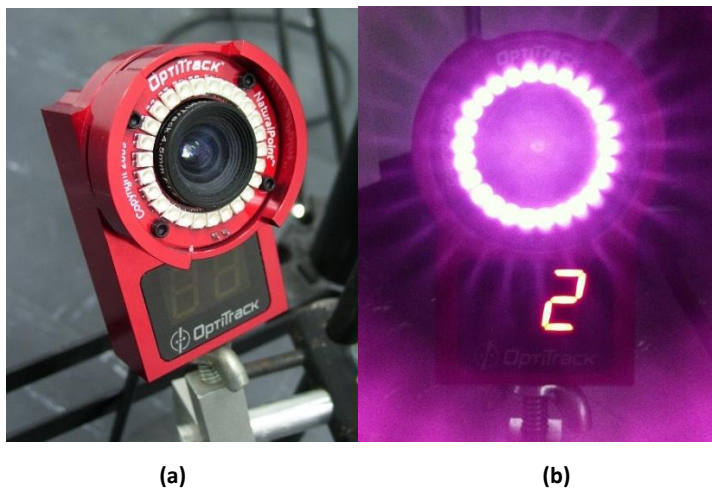


Figura 2-5. (a) Una de las ocho cámaras infrarrojas ubicadas en el laboratorio. (b) Haces infrarrojos emitidos por las cámaras.

Marcadores

Los marcadores utilizados para la detección son básicamente superficies de goma cubiertas con material reflectante (Figura 2-6). Los marcadores más pequeños, normalmente utilizados para análisis de movimientos faciales tienen un diámetro de 3 mm y están cubiertos por una capa de pintura plateada que refleja la luz infrarroja. Otros marcadores de mayor tamaño, empleados para captura de movimientos de cuerpo entero, son esferas de un diámetro aproximado de 16 mm y están cubiertos por una cinta reflectante también de color plateado, especial para reflejar haces infrarrojos [25].

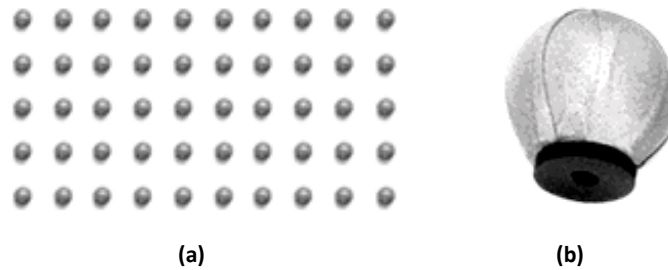


Figura 2-6. (a) Marcadores de 3 mm de diámetro. (b) Marcador para análisis de cuerpo entero. Fuente [26].

Software

El software utilizado para la unión de la información enviada por las cámaras y para la visualización de la nube de puntos se llama Motive® (Figura 2-7). Este software se encarga de la calibración del sistema, determinando las coordenadas espaciales relativas de cada una de las cámaras con respecto a una referencia determinada por el usuario. Esta referencia se define por medio de una escuadra de calibración compuesta por tres puntos que definen el plano de referencia correspondiente al suelo [26].

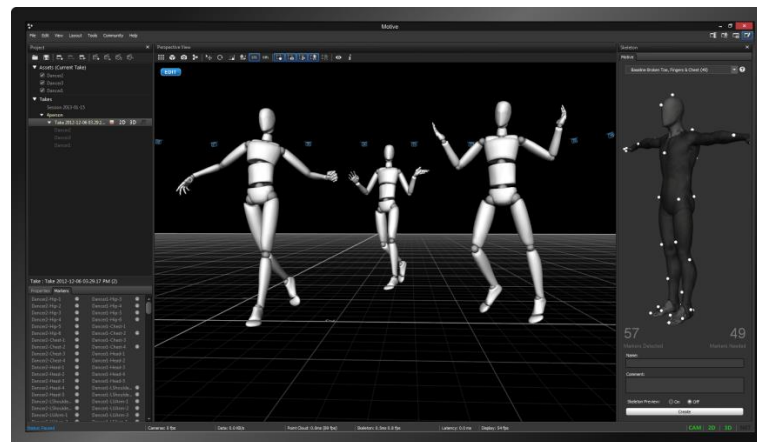


Figura 2-7. Apariencia del software de captura de movimiento. Fuente [26].

Configuración del laboratorio

El Laboratorio de Biomecánica mencionado anteriormente se encuentra diseñado para llevar a cabo pruebas de análisis de cuerpo entero, empleando un traje oscuro de tela suave para adherir a éste marcadores con bases en Velcro™. Dado que este sistema está definido para realizar seguimiento de marcadores de 16 mm de diámetro ubicados por todo el cuerpo, fue

necesario buscar una configuración adecuada para capturar los movimientos descritos por la mano, empleando marcadores de 3 mm. La configuración de cámaras para análisis de la mano difiere sustancialmente de la configuración para cuerpo entero; el volumen de captura es mucho más pequeño, el foco de las cámaras debe centrarse a una altura diferente, las cámaras deben estar más cerca entre ellas y la cercanía entre los marcadores requiere que la ubicación de los dispositivos de captura conforme una geometría también diferente. En una primera aproximación se buscó una configuración de cámaras en una distribución en forma de cubo para realizar una prueba piloto, como se ilustra en la Figura 2-8; para esta configuración se ubicaron las cámaras más bajas a 0,5 m del piso y las cámaras altas a 2,5 m del piso. Luego de varias pruebas se observó que esta ubicación requería realizar tomas de manera independiente por cada dedo, aumentando la duración de las pruebas y dificultando el análisis cinemático en términos de sincronía de los dedos; por lo anterior se propuso una distribución de cámaras semicircular, como se muestra en la Figura 2-15. Esta última configuración permitió capturar de forma simultánea los movimientos de los dedos.

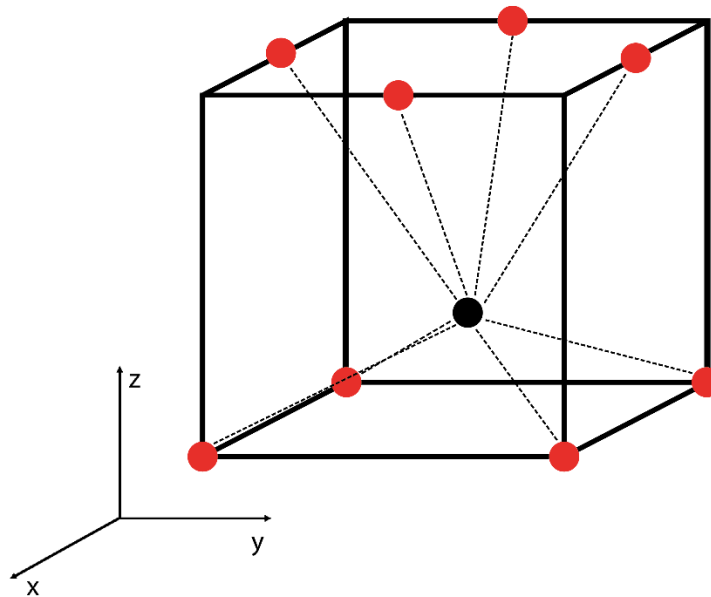


Figura 2-8. Ubicación de las cámaras para la prueba piloto.

2.4 Análisis cinemático

2.4.1 Procesamiento de datos

El sistema descrito en el apartado anterior, exporta los datos capturados para visualizar, analizar o construir animaciones. Dicha exportación es dada en archivos de extensión “.c3d” o “.bvh”, los cuales pueden ser procesados y analizados en Matlab®. Esta investigación se vale de la librería *stickfigure* [27] de libre descarga, para la conversión de archivos de extensión “.c3d”. La información procesada por el *toolbox* mencionado es almacenada en una matriz tridimensional (explicada esquemáticamente en la Figura 2-9) que contiene las coordenadas xyz de cada uno de los marcadores ubicados para la captura de movimiento, donde el número de filas de dicha matriz corresponde al número de *frames* (cuadros) capturados (el sistema tiene una resolución de captura de 100 cuadros por segundo), el número de columnas corresponde al número de marcadores capturados y las tres dimensiones equivalen a los tres ejes coordenados *x*, *y*, *z*. Así pues, si se realiza una captura de 20 marcadores durante cinco segundos, se extraerá una matriz de 500 filas, 20 columnas y tres dimensiones (500x20x3); donde cada dimensión es una matriz 500x20 conteniendo los valores para los ejes *x*, *y*, *z* respectivamente.

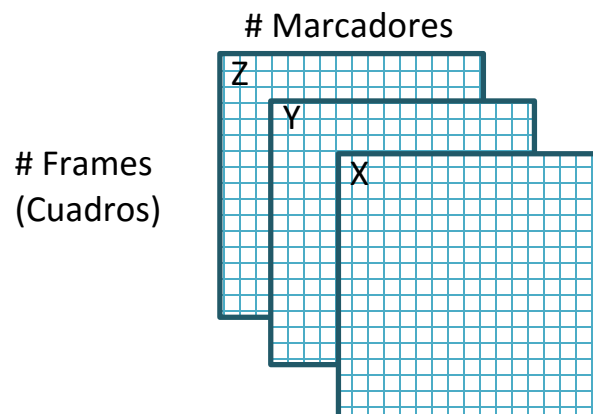


Figura 2-9. Esquema de composición de las matrices tridimensionales empleadas para el análisis.

2.4.2 Pre-procesamiento y variables angulares

El software de captura de movimiento enumera los marcadores de manera aleatoria en cada una de las grabaciones, dificultando la definición de vectores que describen los segmentos que componen el dedo. Esta situación obligó a la creación de un algoritmo para seleccionar y ordenar los marcadores necesarios para diferenciar cada dedo, de tal manera que en la primera columna de la matriz tridimensional se consignaran los datos de la articulación de la muñeca, y así en ese orden hasta dejar para la última columna los datos del extremo distal del dedo.

Con los marcadores seleccionados, reordenados y transformados se procede a calcular los ángulos en cada una de las articulaciones que intervienen en los movimientos de agarre y en las diferentes posturas analizadas. Para este cálculo se acogió el concepto del producto punto, el cual, matemáticamente, define la proyección de un vector en otro, de tal manera que el resultado es el producto de las normas de ambos vectores multiplicado por el coseno del ángulo entre ellos [28]:

$$A \cdot B = |A||B| \cos \theta \quad (1)$$

Para definir el ángulo entre los vectores A y B se define:

$$\theta = \arccos\left(\frac{A \cdot B}{|A||B|}\right) \quad (2)$$

A partir de la expresión anterior se construyó un algoritmo para determinar los ángulos articulares de interés para el análisis de cada dedo en todos los *frames* capturados. Debido a la presencia de ruido asociado a los rayos infrarrojos de las cámaras, que por la naturaleza del sistema de captura es imposible de eliminar, los conjuntos de datos angulares presentaron perturbaciones (Figura 2-10 (a)). Por tal motivo, se aplicó un filtro FIR de suavizado [29] a todos los vectores de ángulos resultantes por articulación (Figura 2-10 (b)).

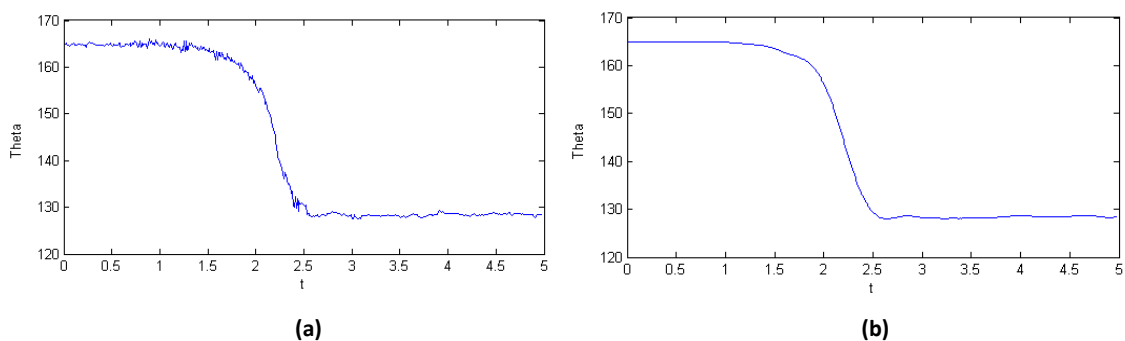


Figura 2-10. Ángulo calculado para la articulación MCF del dedo 2. (a) Antes de aplicar el proceso de filtrado. (b) Luego de aplicar el proceso de filtrado.

2.4.3 Prueba piloto

Inicialmente se realizó una prueba de captura de movimiento piloto con un solo sujeto. Para dicha prueba, con el fin de lograr una mayor precisión en la captura y también para evitar pérdidas, oclusiones o duplicaciones en las mediciones de los marcadores, se optó por analizar independientemente cada dedo en los diferentes movimientos estudiados. Para facilitar la visualización y manipulación de los datos se creó un *script* con el fin de seleccionar y ordenar los marcadores de interés para un dedo en particular. En cada prueba fueron ubicados marcadores en las articulaciones de la muñeca (para medir la articulación carpo-metacarpiana del pulgar - CMC), metacarpo-falángicas e interfalángicas (Figura 2-11).

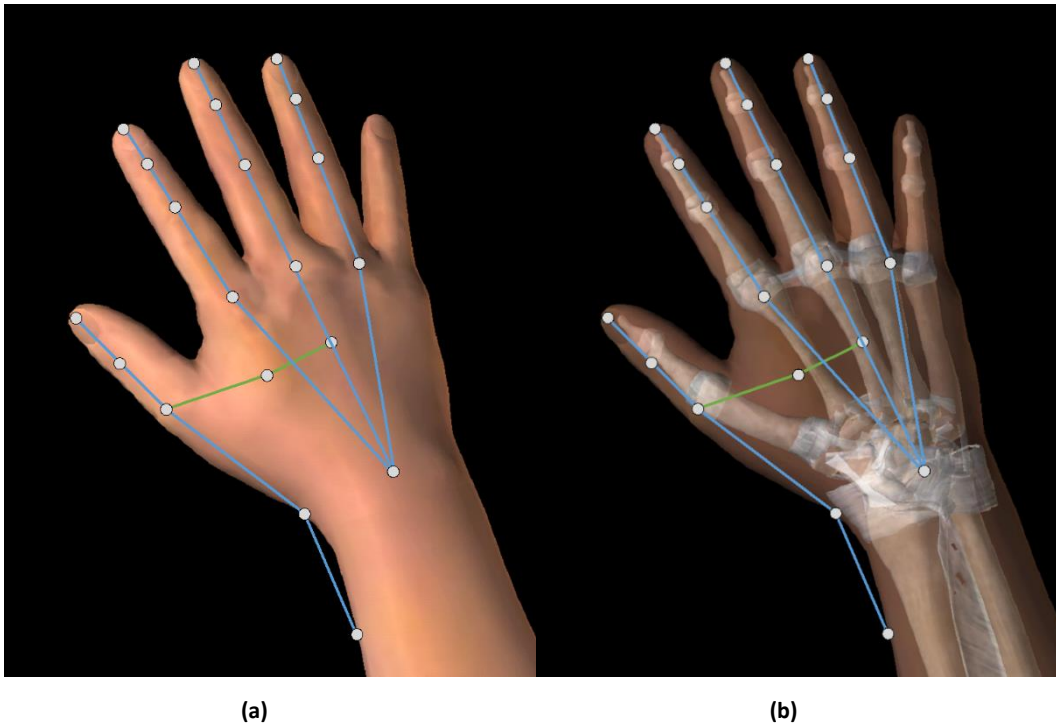


Figura 2-11. Representación de ubicación de marcadores y las rectas formadas por éstos para la definición de ángulos articulares.

Las rectas de color verde definen el ángulo de oposición del pulgar con respecto al dorso de la mano. (a) Ubicación de marcadores sobre la piel. (b) Visualización de la ubicación de los marcadores con respecto a las articulaciones de los dedos.

Luego de ordenar los marcadores desde proximal hasta distal, se procede a trasladar el origen del sistema de coordenadas de la matriz tridimensional resultante de tal manera que el punto $(0,0,0)$ quede determinado en el origen de cada dedo (articulación proximal). Para llevar a cabo lo anterior se empleó el concepto de transformación de sistemas coordenados a partir de la inversa de una matriz de transformación homogénea [30]. Las matrices A_T y ${}^A_T^{-1}$, matriz de transformación homogénea y su inversa, respectivamente, se definen con respecto a un sistema de coordenadas $\{A\}$ de referencia (equivalente al sistema de coordenadas del software Arena®) y un sistema de coordenadas local al dedo $\{B\}$ (centrado en la articulación de la muñeca). La matriz de transformación homogénea se define a continuación:

$${}^A_T = \begin{bmatrix} {}^A_R & {}^A P_{ORGB} \\ 0 & 1 \end{bmatrix} \quad (3)$$

Donde:

- ${}^A_B R$: Es la matriz rotacional del sistema de coordenadas {B} en el sistema de coordenadas {A}
- ${}^A P_{ORGB}$: Es el vector columna que define las coordenadas del origen del sistema {B} expresadas en el sistema {A}.
- 0: Vector fila de tres ceros.
- 1: Unidad agregada junto con el vector de ceros para que la matriz de transformación sea cuadrada y homogénea, y por lo tanto invertible.

Esta matriz de transformación permite expresar las coordenadas del sistema local en el sistema global, pero se busca una representación inversa, es decir, expresar las coordenadas del sistema global en el sistema local. Para lograr lo anterior basta con invertir la matriz de transformación homogénea, lo cual, desde el punto de vista computacional, no es práctico (computacionalmente costoso en tiempo de procesamiento) si se piensa emplear una función de inversión de matrices. Para disminuir tiempos de cómputo se procede a definir la inversa de la matriz a partir de las sub-matrices y vectores que la componen [30]:

$${}^A_B T^{-1} = \begin{bmatrix} {}^A_B R^T & -{}^A_B R^T * {}^A P_{ORGB} \\ 0 & 1 \end{bmatrix} \quad (4)$$

La matriz de rotación ${}^A_B R$ se definió según el ángulo existente entre el segmento metacarpiano y el eje y de referencia. Luego de aplicar la transformación mencionada, cada dedo quedó definido como se muestra en la Figura 2-12.

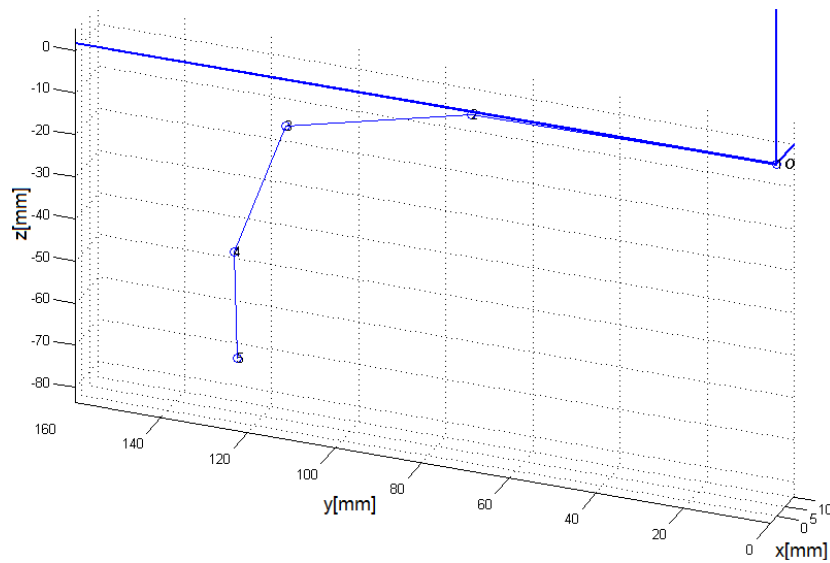


Figura 2-12. Visualización del dedo tras aplicar la transformación de coordenadas.

2.4.4 Captura final

Con el objetivo de obtener un modelo cinemático comparable con el prototipo de prótesis, se definieron los siguientes criterios para los participantes de las pruebas de captura de movimiento, buscando similitud de dimensiones con el prototipo:

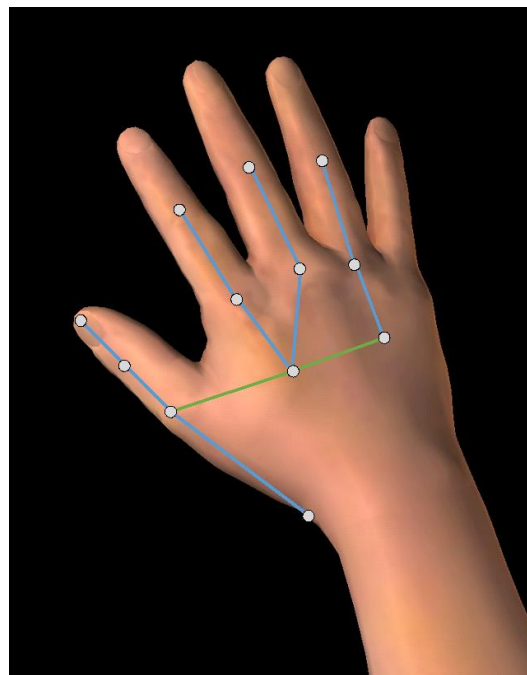
- Poseer una longitud palmar (medida desde la articulación de la muñeca hasta el extremo distal del tercer dedo) de 19 cm con una tolerancia de +/- 1 cm; buscando similitud con el prototipo de prótesis de mano.
- Poseer un perímetro igual o inferior a 7 cm en la falange proximal del tercer dedo.
- Predominancia lateral derecha (diestros).
- No presentar o haber presentado lesiones osteomusculares en el miembro superior derecho.
- No padecer enfermedades que afecten el desempeño neurológico, motriz o muscular.
- Participar voluntariamente en las pruebas de captura de movimiento.

Luego de realizar una encuesta de inspección, se logró contactar a tres individuos que cumplen con los criterios mencionados anteriormente. Esta información se encuentra en la Tabla 2-4.

Tabla 2-4. Medidas de la mano derecha tomadas en los individuos que participaron en la captura de movimiento.

Individuo	Longitud palmar [cm]	Perímetro dedo 3 [cm]
1	19,5	6,9
2	19,4	6,5
3	18,0	7,0

Con las personas contactadas se realizaron diez tomas por cada movimiento (agarres cilíndrico, esférico, plano y pinza), todos partiendo desde una postura de relajación, dando un recuento de 40 grabaciones por sujeto y un total de 120 grabaciones analizadas. Con el fin de transcribir la información de la captura de movimiento a la prótesis de mano, se ubicaron marcadores en las articulaciones MCF (de todos los dedos), CMC, IF y OP, tal y como se muestra en la Figura 2-13. En las sesiones de captura definitivas fue posible grabar y analizar de manera simultánea la información de todos los marcadores. Un ejemplo de la visualización en Matlab® de todos los marcadores se puede observar en la Figura 2-14.

**Figura 2-13. Ubicación de marcadores en las pruebas finales.**

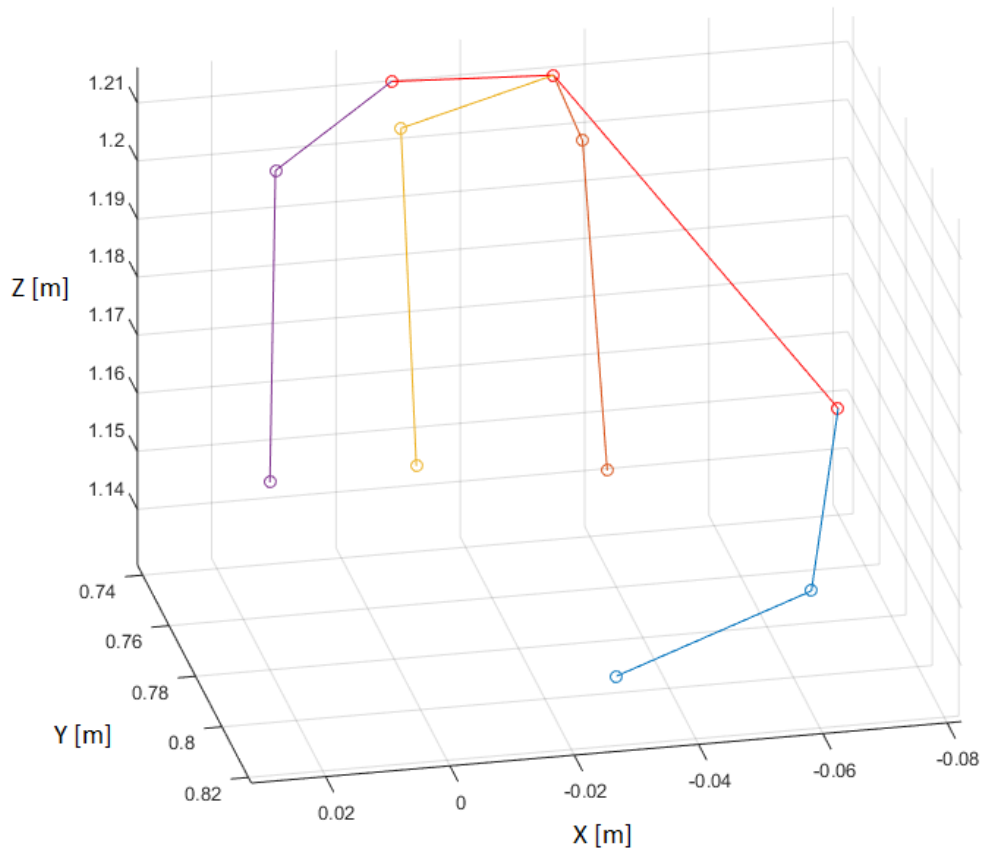


Figura 2-14. Visualización en perspectiva en Matlab® de los marcadores ubicados en la mano.

Para contrastar la información de los tres individuos se decidió analizar únicamente la fase de cierre, tomando como variables determinantes los ángulos inicial y final, el tiempo requerido para ejecutar el cierre y la pendiente de cierre o apertura. Es necesario señalar, que para algunas articulaciones y ciertos movimientos, no fue analizada la pendiente dado que el rango de movimiento era considerablemente pequeño.

En las pruebas de captura de movimiento finales se definió una posición de cámaras diferente a la presentada para la prueba piloto, con el fin de visualizar todos los dedos simultáneamente. En general, se busca que las ocho cámaras apunten sus focos hacia el volumen en el cual se desarrollará el movimiento (agarres en la mano), intentando observar todos los ángulos posibles, evitando que las cámaras se vean entre ellas. En la Figura 2-15 se puede apreciar esta configuración.

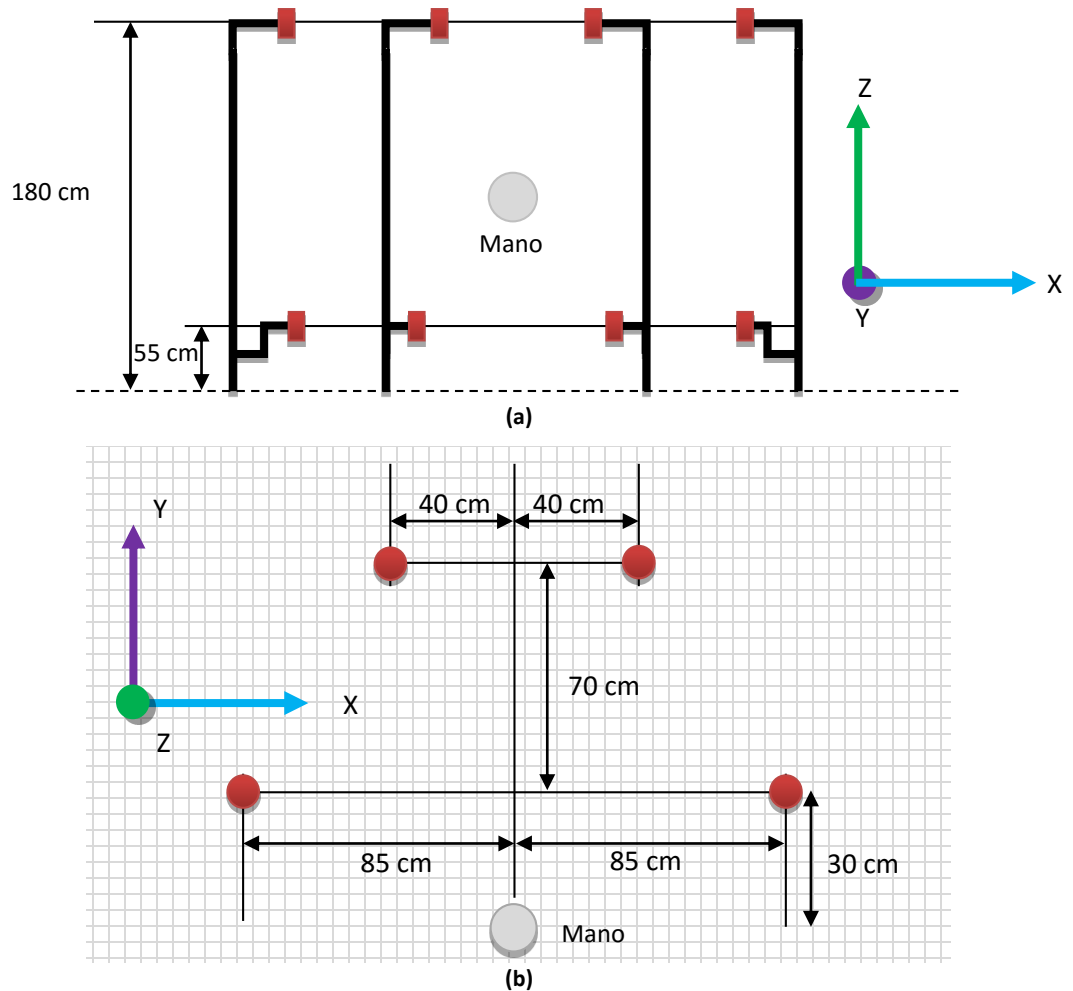


Figura 2-15. Distribución final de las cámaras. (a) Vista frontal. (b) Vista superior.

Luego de establecer la ubicación de las cámaras y los demás parámetros necesarios para llevar a cabo las pruebas de captura de movimiento, se contactaron tres personas con dimensiones palmares similares a las del prototipo de prótesis (longitud palmar medida desde la base de la palma de la mano hasta el extremo distal del dedo 3). Cada individuo ejecutó acciones de agarre cilíndrico, esférico, plano y de pinza bidigital, utilizando siempre la misma esfera, el mismo cilindro y la misma lámina de balsa. Los resultados de estas pruebas se consignan en la Sección 4.2.

3. Control

En el presente Capítulo se explica la implementación algorítmica del control del prototipo de prótesis de mano emmpleando tres etapas: la primera etapa consiste en la calibración y filtrado de las señales arrojadas por los sensores de flexión para ser interpretadas como ángulos de las diferentes articulaciones del prototipo. En la segunda etapa se presenta el desarrollo de los algoritmos necesarios para el control individual de cada dedo en lazo abierto mediante sliders en una interfaz gráfica en Matlab®, para verificación de la medida entregada por los sensores y calibración de la histéresis. La última etapa culmina el proceso de control, en la cual se implementa un sistema de control en lazo cerrado por tipos de agarre, utilizando comandos de voz como entradas de accionamiento.

3.1 Calibración de Sensores

Los sensores de flexión resistivos, mencionados en el Anexo A que se encuentra más adelante, cambian su resistencia eléctrica dependiendo del grado de flexión que presentan, de tal manera que a mayor ángulo de flexión, mayor es la resistencia eléctrica del mismo [31]. Con el fin de obtener una señal voltaica que permita controlar cada uno de los grados de libertad que presenta el prototipo de prótesis, cada sensor fue conectado en serie con una resistencia de 10 k Ω (un pin del sensor conectado a 5V y un pin de la resistencia fija a tierra), llevando el divisor de voltaje a un búfer de impedancia [32] con el fin de asegurar un nivel de tensión estable.

Cada señal de los sensores es enviada a pines de conexión análoga, es decir, Conversores Análogo-Digital (ADC), de una tarjeta Arduino® Mega 2560. Cada ADC cuenta con una resolución de 10 bits (1024 valores diferentes) convirtiendo cada señal de entrada (entre 0 y 5V) a un número entero (entre 0 y 1023 bits), donde cada bit equivale a aproximadamente 4,9 mV [33].

Se construyó una interfaz gráfica en Matlab® con el fin de leer los valores en bits de cada uno de los sensores de flexión y almacenar los mismos para una posible calibración. El código de dicha interfaz se encuentra en el Anexo: Códigos escritos en el entorno MATLAB® y su apariencia se muestra en la Figura 3-1. Cada GDL se activa seleccionando uno de tres *radiobuttons*, donde cada uno permite extender, flexionar o detener el movimiento de una articulación. A la izquierda de la interfaz se pueden observar dos espacios para cada sensor de flexión (se adicionaron tres espacios más para añadir sensores de fuerza), permitiendo ver en uno de dichos espacios el valor en bits de cada sensor y dejando el otro para una posterior conversión de dicho valor a un número angular en grados.

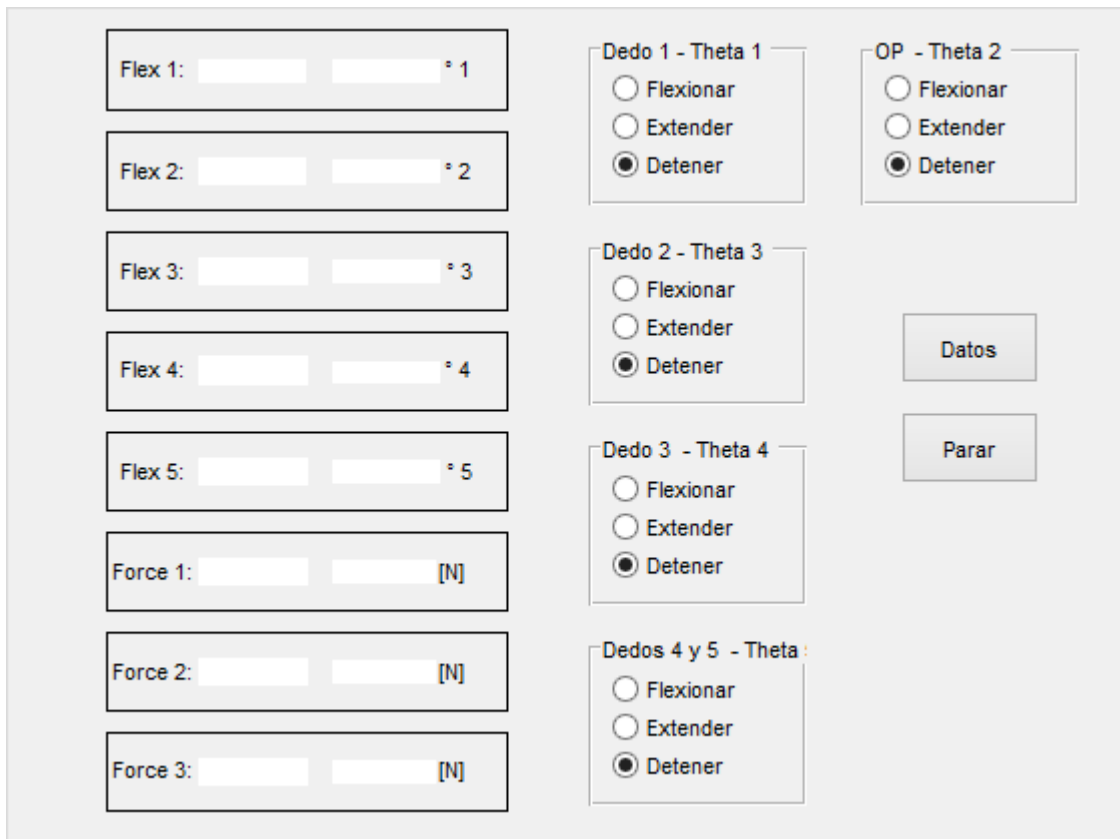


Figura 3-1. Interfaz de accionamiento para visualización de ángulos en la prótesis.

Para relacionar la medida del sensor con un ángulo de flexión, se realizó una captura de movimiento con cámaras infrarrojas, ubicando tres marcadores en cada sensor, igual a lo realizado en las sesiones de captura de movimiento (Capítulo 2), asegurando que se posicionara un marcador exactamente sobre el vértice de la articulación y los otros dos en los dos segmentos

móviles adyacentes a cada articulación. Esta captura de datos se ejecutó directamente en el prototipo de prótesis, cubriéndola con un guante de látex para simular la piel y así el deslizamiento de los sensores reflectivos en la misma. La ubicación de marcadores se puede apreciar en la Figura 3-2. Cabe recalcar que la ubicación de los marcadores resultó ser muy similar a la expuesta en el capítulo anterior (Figura 2-11).



Figura 3-2. Ubicación de marcadores en el Prototipo de Prótesis.

Dado que el sistema de cámaras infrarrojas presenta una tasa de muestreo de 100 cuadros por segundo (100 coordenadas por segundo y por ende 100 ángulos calculados por cada segundo) [3], es decir, 100 Hz, y el sistema de captura de señales eléctricas (Transmisión desde Arduino® a Matlab®) presenta una tasa de muestreo de aproximadamente 30 muestras por segundo [34], es decir, 30 Hz, fue necesario seleccionar un sub-muestreo [35] de los ángulos arrojados por el sistema de captura de movimiento para que pudiesen ser comparados con los datos entregados por el ADC de Arduino®. Siendo $x[n]$ el vector original de ángulos calculados y $y[n]$ el vector de ángulos muestreados, el sub-muestreo se define así:

$$y[n] = x[3,3 * n] \quad (5)$$

Luego de realizar el sub-muestreo, por medio de MS Excel se construyeron gráficas del ángulo calculado vs el valor entregado por el sensor, para luego encontrar una curva de ajuste a través de la función 'Línea de tendencia'. Los grados de libertad del prototipo fueron enumerados de la siguiente manera:

- θ_1 : ángulo de la articulación IF del dedo 1
- θ_2 : ángulo de oposición (OP)
- θ_3 : ángulo de la articulación MCF del dedo 2
- θ_4 : ángulo de la articulación MCF del dedo 3
- θ_5 : ángulo de la articulación MCF del dedo 4 (para control de los dedos 4 y 5)

Desde la Figura 3-3 hasta la Figura 3-7 se muestran gráficas extraídas de MS Excel.

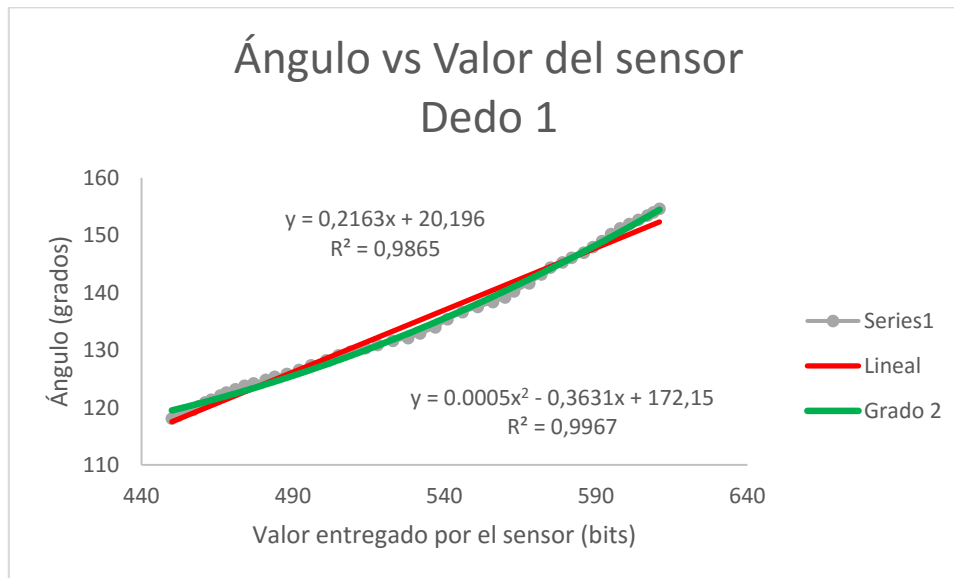


Figura 3-3. Ángulo calculado contra el valor entregado por el sensor para theta 1.

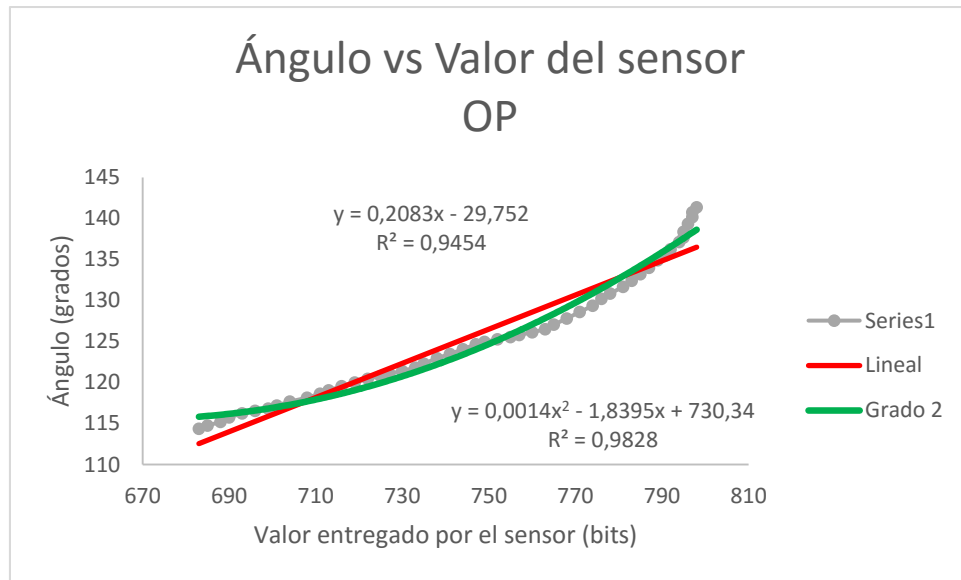


Figura 3-4. Ángulo calculado contra el valor entregado por el sensor para theta 2.

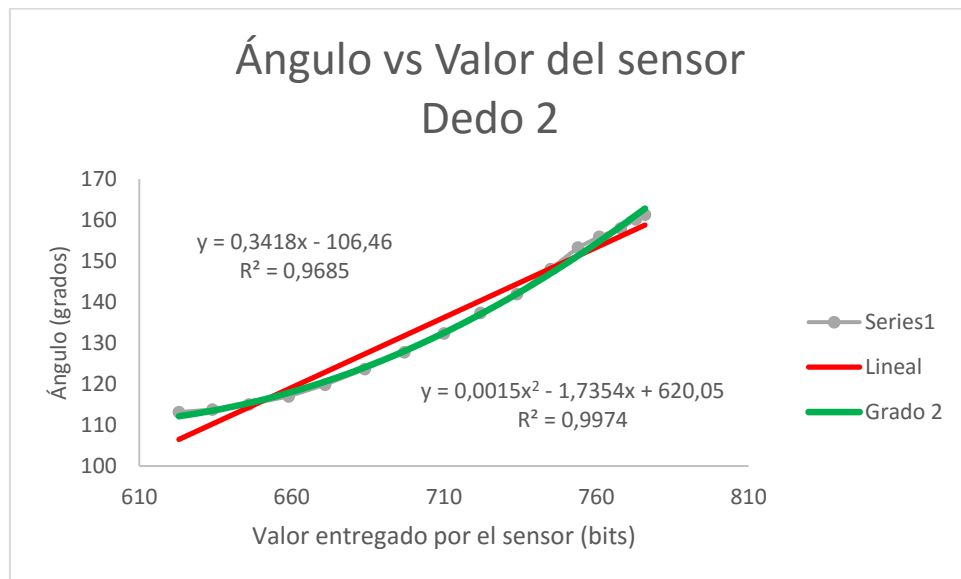


Figura 3-5. Ángulo calculado contra el valor entregado por el sensor para theta 3.

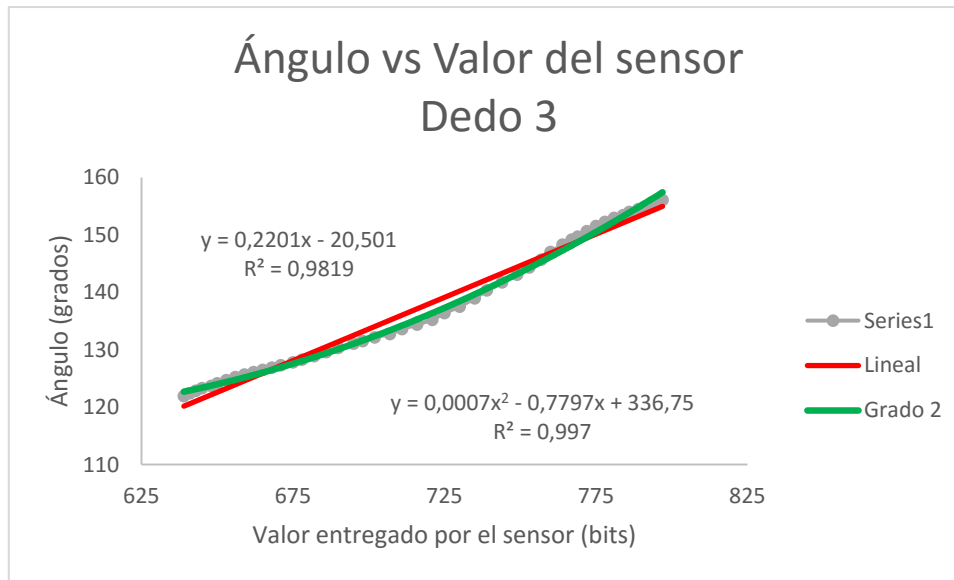


Figura 3-6. Ángulo calculado contra el valor entregado por el sensor para theta 4.

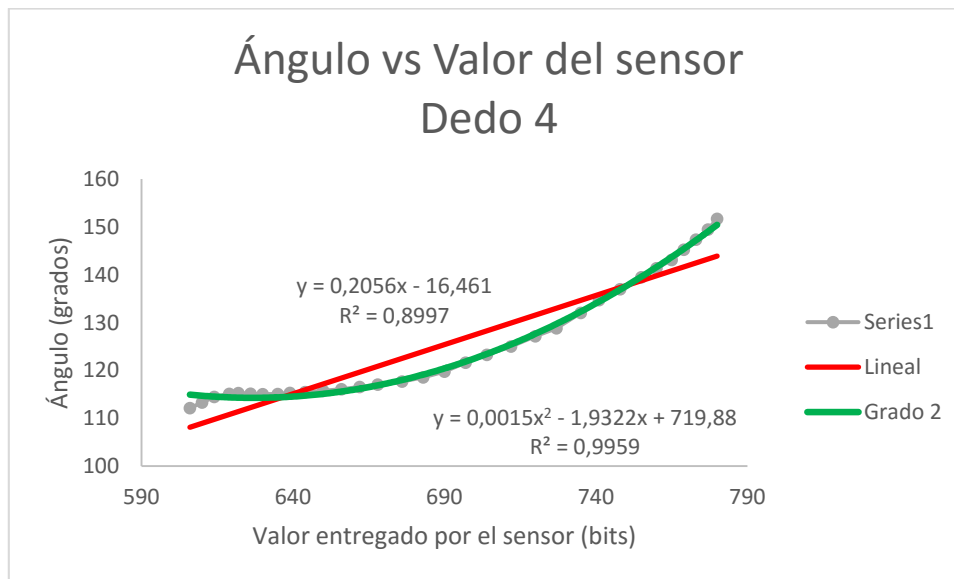


Figura 3-7. Ángulo calculado contra el valor entregado por el sensor para theta 5.

En las gráficas se puede observar que la curva que mejor se ajusta para convertir el número en bits a un ángulo es de grado 2 para todos los sensores. A partir de lo anterior surgen las siguientes ecuaciones de ajuste:

$$\theta_1 = 0,0005f_1^2 - 0,3631f_1 + 172,5 \quad (6)$$

$$\theta_2 = 0,0014f_2^2 - 1,8395f_2 + 730,34 \quad (7)$$

$$\theta_3 = 0,0015f_3^2 - 1,7354f_3 + 620,05 \quad (8)$$

$$\theta_4 = 0,0007f_4^2 - 0,7797f_4 + 336,75 \quad (9)$$

$$\theta_5 = 0,0015f_5^2 - 1,9322f_5 + 719,88 \quad (10)$$

Donde f_1, f_2, f_3, f_4 y f_5 son los valores en bits entregados por los respectivos sensores de flexión para cada articulación.

3.2 Control en Lazo Abierto

Con el fin de controlar el ángulo de cada uno de los dedos se construyó una segunda interfaz gráfica en Matlab® para accionar cada GDL del prototipo. La estrategia abordada fue la de un control proporcional por histéresis [36], con una constante de proporcionalidad máxima para cada motor y un rango de decisión de +/- 5°; esto quiere decir que cada micromotor es controlado por una señal PWM a un 100% del ciclo de trabajo [37] para lograr la mayor velocidad permitida por sus características electrónicas (5V como voltaje máximo de alimentación), y que el criterio de parada de la articulación se da si se encuentra en un rango de +/- 5° alrededor del ángulo deseado (medido por los sensores de flexión).

El control en lazo abierto fue implementado con el fin de verificar que las curvas de ajuste de los sensores fueran las adecuadas para cada GDL; por este motivo, únicamente se emplea esta estrategia de control con la prótesis en "vacío", es decir, sin agarrar objetos. En la Figura 3-8 se ilustra la apariencia de dicha interfaz y su código se encuentra en el Anexo: Códigos escritos en el entorno MATLAB®. Al dar clic en el botón "Conectar", cada *slider* toma el valor medido por su respectivo sensor de flexión al momento de en lazar Matlab® con Arduino® con el fin de sincronizar la interfaz con el estado actual de la prótesis. Al manipular cada *slider*, en cada

espacio de texto aparecerá el ángulo definido por la interfaz (deseado) y el ángulo logrado en la prótesis.

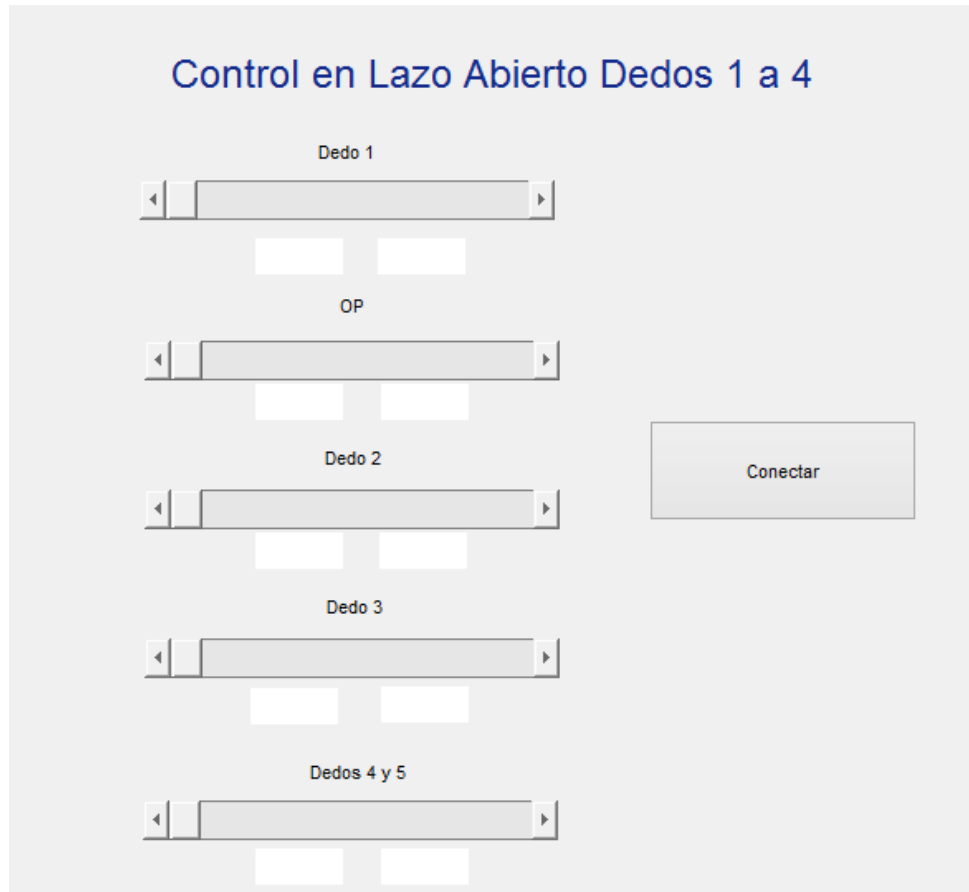


Figura 3-8. Interfaz gráfica para control en lazo abierto de la prótesis.

3.3 Control de Movimientos por Agarres y Posturas Empleando Comandos de Voz

Para lograr un sistema de control por comandos de voz se eligió como estrategia de detección de dichos comandos el uso de Redes Neuronales Artificiales (RNA o ANN en inglés). Esta metodología consiste en construir un algoritmo compuesto por neuronas que se entrenan para tomar decisiones en función de unos pesos y un error deseado. A continuación se explica más a fondo esta metodología.

Redes neuronales

Las redes neuronales artificiales (RNA o ANS por sus siglas en inglés) son sistemas de procesamiento automático inspirado en la forma en que funciona el sistema nervioso con el fin de construir un sistema de información paralela, distribuido y adaptativo que pueda presentar un comportamiento inteligente.

Estructura general de un sistema neuronal artificial (ANS)

Una red neuronal artificial copia de cierta forma la estructura de una red neuronal biológica con el fin de alcanzar una funcionalidad y resultados similares [38].

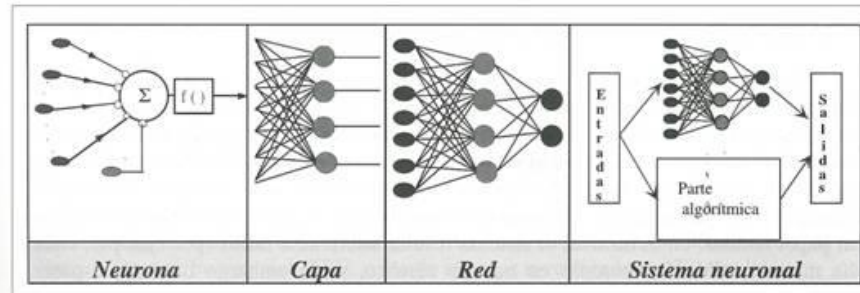


Figura 3-9. Estructura jerárquica de un sistema basado en ANS [38].

Se define neurona a un dispositivo simple de cálculo que, partiendo de un vector de entrada que contiene la información del exterior, arroja una sola salida. Los elementos que constituyen una neurona son:

- Conjunto de entradas: Pueden ser digitales o análogas.
- Pesos sinápticos: Densidad de interacción entre neuronas, es decir entre una pre-sináptica (j) y una post-sináptica (i).
- Regla de propagación: Calcula a partir de las entradas y los pesos de las neuronas el valor del potencial post-sináptico de la neurona. Se interpreta como el producto escalar de los vectores de entrada y los pesos
- Función de activación: O función de transferencia, proporciona el estado de activación actual a partir del potencial post-sináptico y del estado de activación anterior. Define por completo la neurona [38].

- **Función de salida:** Proporciona la salida de la neurona dependiendo de su función de activación.

Arquitectura de redes neuronales

Es la forma en la que todos los componentes de la red neuronal se asocian. Las neuronas se agrupan en unidades estructurales denominadas capas (Figura 3-10). El conjunto de una o más capas constituye una red neuronal [38].

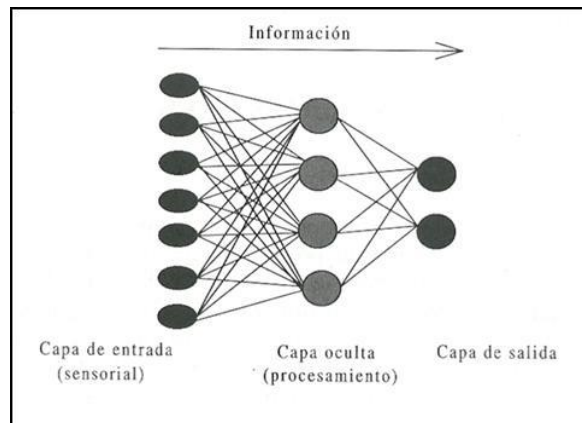


Figura 3-10. Capas de una Red Neuronal Artificial [38].

Se pueden diferenciar tres capas:

- **Capa de entrada:** Neuronas que reciben información del exterior.
- **Capa oculta:** Neuronas que no tienen contacto con el exterior y se encargan de procesar la información.
- **Capa de salida:** Neuronas que muestran el resultado final de todo el procesamiento de la red.

Tipos de Redes Neuronales

- **Red Neuronal monocapa:** Es la red neuronal más sencilla ya que se tiene una capa de neuronas que proyectan las entradas a una capa de neuronas de salida.
- **Red Neuronal multicapa:** Existe un conjunto de capas intermedias entre las capas de entrada y de salida (neurona oculta).
- **Red Neuronal recurrente:** Los lazos pueden ser entre neuronas de diferentes capas, neuronas de la misma capa o, entre una misma neurona.

Aprendizaje de una red neuronal

“El aprendizaje es un proceso por el cual los parámetros libres de una red neuronal son ajustados a través de un proceso continuo de estimulación por parte del entorno en donde se sitúa el sistema. El tipo de aprendizaje es determinado por la forma que tienen lugar dichos cambios” [39]. En otras palabras, el aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada. Los cambios que se producen durante el mismo se reducen a la destrucción, modificación y creación de conexiones entre las neuronas [40].

Hay dos métodos de aprendizaje importantes que pueden distinguirse, el aprendizaje supervisado y el no supervisado. Para el caso particular de esta investigación se emplea el primero:

- a) **Aprendizaje supervisado:** Se realiza mediante un entrenamiento controlado por un agente externo (usuario), que determina las respuestas de la red a partir de una entrada determinada.
- b) **Aprendizaje por corrección del error:** Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red, es decir, en función del error cometido en la salida.

Desarrollo del algoritmo para detección de comandos

En la Figura 3-11 se muestra la arquitectura de la RNA requerida para llevar a cabo el control en lazo cerrado. Esta red se compone por los siguientes elementos:

- Z: Neuronas de la capa de salida, corresponden a los coeficientes *LPC* (predictores lineales) (son 13).
- V: Pesos sinápticos de las neuronas de la capa de salida a la capa oculta.
- Y: Neuronas de la capa oculta (son 12).
- W: Pesos sinápticos de las neuronas de la capa oculta a la capa de salida.

- O: Neuronas de salida (son seis y corresponden a cada comando a detectar: abrir, cerrar, pinza, bien y señalar).

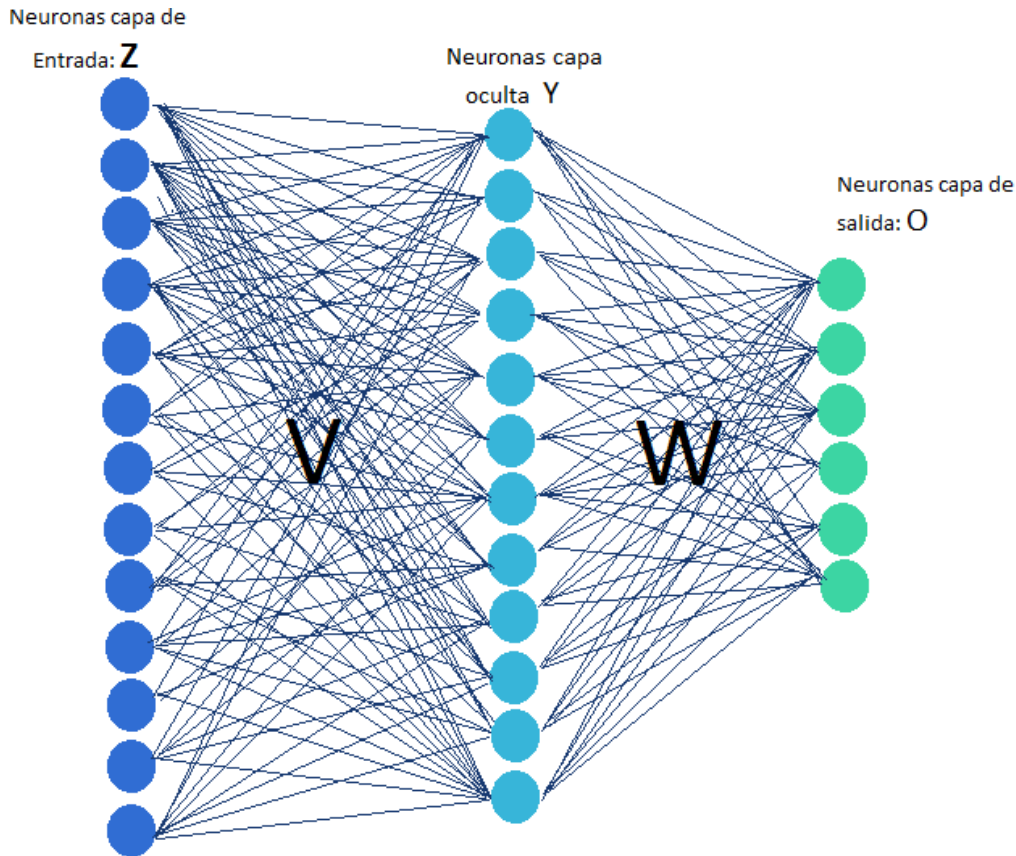


Figura 3-11. Arquitectura red neuronal para control en lazo cerrado.

La red neuronal artificial aplicada en este proyecto es de retro-propagación o *backpropagation* (o propagación del error hacia atrás), la cual funciona con el aprendizaje predefinido de pares entrada-salida, es decir, primero se genera una entrada que estimule la primera capa de la red; este estímulo se propaga a través de las demás capas hasta que genera una salida. Este resultado se compara con la salida que desea obtenerse y se calcula un valor de error para cada salida [41]. Dicho proceso se replica partiendo de la capa de salida hacia todas las neuronas de la capa oculta, repitiéndose de forma que todas las neuronas de la red hayan recibido una señal de error.

El proceso de propagación hacia atrás es importante porque a medida que se entrena la red, todas las neuronas de las capas intermedias se organizan de modo tal que las distintas neuronas reconozcan distintas características de la capa de entrada de forma que si una nueva entrada contiene un patrón que encaje con las características previamente aprendidas, esta entrada será reconocida y clasificada.

Para el desarrollo de la interfaz gráfica en el software Matlab®, se adaptó el algoritmo desarrollado por [42] originalmente escrito para el sistema de desarrollo *Mbed*®, para control de un robot que es accionado por dos motores. El funcionamiento del código se muestra esquemáticamente en la Figura 3-12.

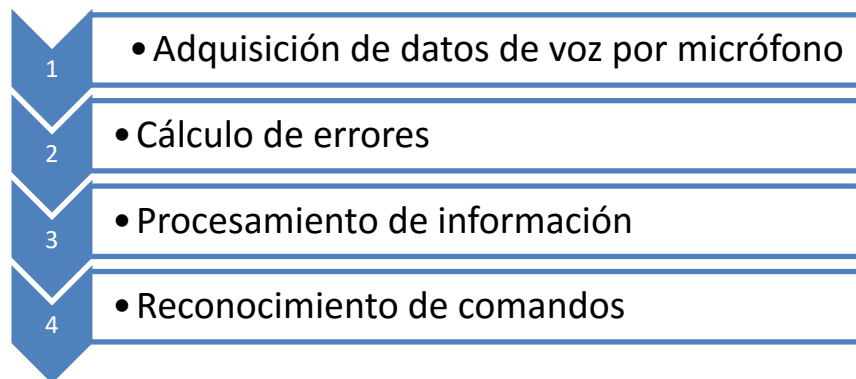


Figura 3-12. Esquema de funcionamiento de la red neuronal que permite el reconocimiento de los comandos de voz.

Según la estructura mostrada en la Figura 3-12, se creó una interfaz gráfica a partir de diferentes adaptaciones al código original tales como: aumentar el número de iteraciones del cálculo del error a 20000, cambiar los comandos a reconocer (Pinza, Abrir, Cerrar, Bien y Señalar); y finalmente en la fase de reconocimiento, mostrar una imagen ilustrando la acción seleccionada por el usuario junto con el mensaje que indica el nombre de la acción. El código puede verse en el Anexo: Códigos escritos en el entorno MATLAB®.

Para la etapa de grabación y entrenamiento de la red neuronal se empleó el comando “*audiorecorder*” el cual se encarga de grabar un sonido. En el presente trabajo se emplean los siguientes parámetros de entrada: frecuencia de muestreo (en este caso de 8000), número de Bits (8) y el número de canal (1) con un tiempo de grabación de 1 segundo. Luego se crea un

objeto extrayendo las características previamente almacenadas en la variable *recorder*. Este proceso se repetirá 10 veces por cada comando, lo cual deja un total de 60 muestras.

Una vez creado el objeto, este entra en una función LPC junto con el número de características (12+1), aplicándole un filtro FIR con el fin de minimizar el proceso de predicción del error, para finalmente almacenarlo en una variable “s” que corresponderá a las neuronas de entrada de la red neuronal. Luego se encuentra a la etapa del cálculo del error relativo, el cual se obtiene realizando el siguiente proceso:

1. Se crea una matriz cuadrada de 6x6 o matriz de etiquetas. Cada fila de la matriz corresponde a un comando.
2. Se concatena 10 veces la matriz de etiquetas, resultando 60 etiquetas cada una de 6x6. Cada una de ellas corresponde a una muestra para el entrenamiento.
3. Se inicializa el peso W mediante una matriz aleatoria normal, cuyas características sean el número de neuronas de salida y el número de neuronas ocultas.
4. Se inicializa el peso V mediante una matriz aleatoria, cuyas características sean el número de neuronas ocultas y el número de neuronas de entrada.
5. Se aplica la función de activación (Figura 3-13), en este caso se eligió la función sigmoidea dado que los valores máximos y mínimos de los coeficientes LPC se encuentran entre -1 y 1.
6. Se calcula el error en cada nodo o muestra por comandos de salida según:

$$Error_{nodo} = \frac{1}{2} (|salida esperada - salida real|^2 + error_{nodo}) \quad (11)$$

7. Se calcula un delta de error para cada neurona de salida según,

$$Delta(k, :) = \frac{1}{2} (Salida esperada(k) - Salida real(k) * (1 - salida real(k)^2)) \quad (12)$$

8. Se calcula la señal de error para cada neurona de la capa oculta mediante y la capa de salida,

$$Error = \sum_1^j \left(\sum_1^k Delta(k) * W(k, j) \right) \quad (13)$$

9. Se calculan los pesos finales de salida, $Delta_{yj}$

$$Delta_{yj} = \frac{1}{2} (1 - y(j)^2) * Error \quad (14)$$

10. Finalmente, se ajustan los pesos de salida con el fin de minimizar el error total (W y V),

$$W = W + eta * Delta \quad (15)$$

$$V = V * eta * Delta_{yj} \quad (16)$$

*Eta: Factor de entrenamiento = 0,1

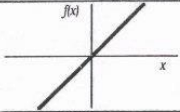
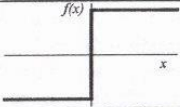
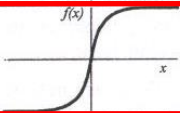
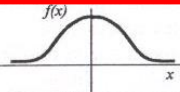
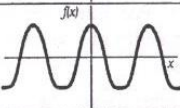
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = \text{sign}(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Sigmoidea	$y = \frac{1}{1 + e^{-x}}$ $y = \text{tgh}(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0, +1]$	
Sinusoidal	$y = A \text{sen}(\omega x + \varphi)$	$[-1, +1]$	

Figura 3-13. Funciones de activación. La elegida es la función sigmoidea [38].

Control en Lazo Cerrado

Para llevar a cabo el control en lazo cerrado, se construyó una tercera interfaz gráfica en Matlab® que, a través de la tarjeta de sonido del PC y un micrófono estándar, permitiese entrenar la RNA almacenando los diferentes comandos, y posteriormente facilitara la recepción del comando para ejecutar la acción. Para cada acción se definen ángulos mínimos finales deseados (en vacío) por cada dedo; y bajo la misma lógica de control proporcional por histéresis empleada en el control en lazo abierto, se busca igualar el ángulo medido con el deseado.

Teniendo en cuenta que la idea es manipular objetos y por consiguiente no siempre se llegará al ángulo mínimo (en vacío), se propone una estrategia de parada por desviación estándar del ángulo medido. Los ángulos medidos por los sensores se almacenan constantemente en una matriz a la cual se puede acceder en cualquier momento durante la ejecución del algoritmo. Al observar una

ventana dinámica de dicha matriz se podrán analizar los últimos datos capturados y entre ellos encontrar diferencias significativas. La metodología propuesta consiste en hallar la desviación estándar [43] de los últimos 30 datos detectados por cada sensor, duplicando la frecuencia de muestreo y así decidir en aproximadamente medio segundo, buscando cuándo ésta se acerca a un valor mínimo que indicará que no existe cambio en la medida; esto se puede interpretar como que el dedo o su articulación no se está moviendo. Si la articulación no presenta cambios, esto se debe, entre otras cosas, a la existencia de un obstáculo (que podría ser el objeto a manipular), y por consiguiente se da la orden de detener el movimiento.

En la Figura 3-14 se muestra la apariencia de la interfaz gráfica para control en lazo cerrado del prototipo y entrenamiento de la RNA, cuyo código se encuentra en el Anexo: Códigos escritos en el entorno MATLAB®. La interfaz cuenta con un botón para iniciar el entrenamiento de la Red Neuronal (Entrenar), un botón para reconocer comandos (Escuchar), un botón para conectar Matlab® y Arduino® (Conectar) y por último un botón para enviar los datos a la prótesis (Mover).



Figura 3-14. Apariencia de la interfaz gráfica para control por voz.

4. Resultados

4.1 Prueba Piloto

Luego de extraer la información angular de cada una de las articulaciones, fue posible identificar tres fases de movimiento descritas en la mayoría de los cierres de los dedos al realizar acciones de agarre (Figura 4-1). La primera fase comprende los movimientos descritos desde que se inicia la grabación hasta el momento en que se hace contacto con el objeto. La segunda fase describe el agarre del objeto desde el instante en que se hace contacto hasta el momento en que se logra un agarre completo. La última fase se puede definir como una etapa de sostenimiento en la que mantiene constante el cierre de los dedos y en la que se afianza el agarre del objeto.

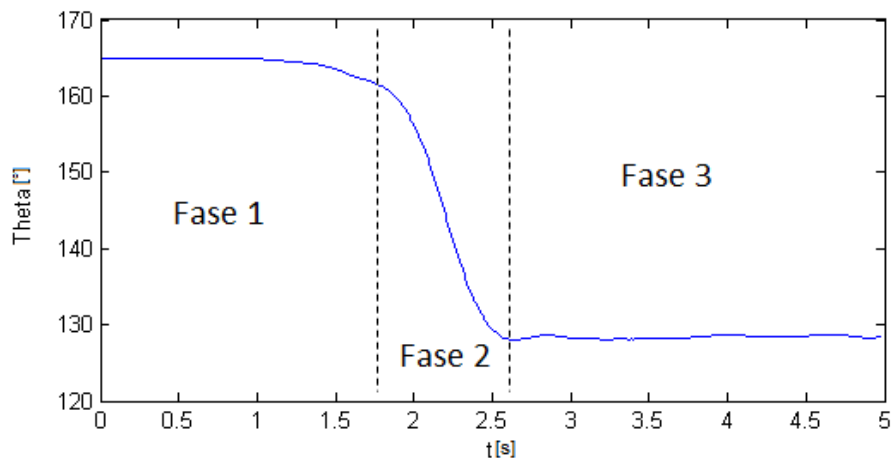


Figura 4-1. Representación gráfica de las fases de acercamiento (1), cierre (2) y sostenimiento (3) presentadas en la mayoría de las articulaciones de la mano.

En la prueba piloto se capturaron movimientos para los agarres cilíndrico, esférico, plano y pinza bidigital y para la postura de mano abierta. A continuación se muestra la información obtenida para los dedos 1, 2, 3 y 4, y para cada tipo de agarre (Tabla 4-1 a Tabla 4-18), indicando la pendiente de cierre o apertura y los respectivos valores en x e y ($X1$, $Y1$, $X2$ y $Y2$) para calcular dicha pendiente; también se encuentran los valores angulares máximos y mínimos alcanzados

por cada articulación. El dedo 5 no se incluyó dado que en el prototipo de prótesis este dedo se acciona simultáneamente con el dedo 4.

Tabla 4-1. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo anular en agarre cilíndrico.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	1,22	167,3	1,88	126,5	-61,82	167,3	121,6
IFP	1,34	148,9	2,04	104,1	-64,00	151,3	103,4
IFD	1,53	171,2	2,13	147,5	-39,50	174,1	145,2

*MCF: Metacarpo-Falángica. IFP: Interfalángica Proximal. IFD: Interfalángica Distal [14].

Tabla 4-2. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo corazón en agarre cilíndrico.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	1,15	157,4	1,78	121,3	-57,30	158,1	119,4
IFP	1,24	159,2	1,87	114,9	-70,32	170,0	111,0
IFD	1,42	173,1	2,09	142,0	-46,42	178,2	140,6

Tabla 4-3. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo índice en agarre cilíndrico.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	1,62	150,8	2,20	130,3	-35,34	154,0	129,0
IFP	1,66	158,8	2,38	108,3	-70,14	163,1	106,0
IFD	1,72	179,1	2,43	148,0	-43,80	179,5	147,8

Tabla 4-4. Ángulos máximos y mínimos y definición de la pendiente de cierre en las cuatro articulaciones del dedo pulgar en agarre cilíndrico.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
CMC	1,62	131,1	2,12	123,5	-15,20	132,9	121,8
MCF	-	-	-	-	-	162,5	160,6
IF	1,86	162,7	2,59	141,2	-29,45	166,9	139,8
OP	1,41	131,7	1,98	118,8	-22,63	133,0	117,0

Tabla 4-5. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo anular en agarre esférico.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	1,55	168,1	1,91	162,7	-15,00	169,2	161,6
IFP	1,57	153,0	2,03	125,7	-59,35	158,7	123,2
IFD	1,58	176,3	2,02	158,0	-41,59	179,5	156,0

Tabla 4-6. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo corazón en agarre esférico.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	1,91	156,0	2,34	148,8	-16,74	156,9	148,4
IFP	1,91	159,3	2,37	134,0	-55,00	164,9	128,0
IFD	1,98	170,0	2,43	156,8	-29,33	174,4	154,7

Tabla 4-7. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo índice en agarre esférico.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	1,41	149,6	1,8	143,3	-16,15	150,1	142,7
IFP	1,51	152,3	2,0	130,0	-45,51	154,4	128,6
IFD	1,65	173,6	2,0	158,9	-42,00	179,0	156,5

Tabla 4-8. Ángulos máximos y mínimos y definición de la pendiente de cierre en las cuatro articulaciones del dedo pulgar en agarre esférico.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
CMC	1,2	134,9	2,05	128,0	-8,12	136,3	127,4
MCF	2,0	158,4	2,43	161,6	7,44	163,3	158,2
IF	2,99	164,4	2,52	138,0	56,17	168,5	136,7
OP	1,13	128,8	1,95	120,4	-10,24	129,4	119,1

Tabla 4-9. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo anular en agarre plano.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	0,59	166,7	1,54	134,1	-34,32	170,8	131,7
IFP	1,23	165,7	1,62	172,7	17,95	173,3	160,6
IFD	-	-	-	-	-	175,5	173,5

Tabla 4-10. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo corazón en agarre plano.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	0,88	150,3	1,31	127,1	-53,95	157,2	123,2
IFP	-	-	-	-	-	179,6	171,0
IFD	-	-	-	-	-	176,9	175,5

Tabla 4-11. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo índice en agarre plano.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	0,72	146,7	1,47	122,8	-31,87	148,3	117,9
IFP	0,87	161,4	1,34	172,8	24,26	174,3	159,6
IFD	-	-	-	-	-	177,6	171,9

Tabla 4-12. Ángulos máximos y mínimos y definición de la pendiente de cierre en las cuatro articulaciones del dedo pulgar en agarre plano.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
CMC	1,35	157,2	1,85	150,4	-13,60	160,7	147,4
MCF	1,51	160,8	1,88	155,7	-13,78	162,8	154,4
IF	1,52	168,9	1,91	171,2	5,90	172,6	166,8
OP	1,40	123,8	1,81	115,6	-20,00	131,9	111,5

Tabla 4-13. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo índice en agarre en pinza.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	0,94	117,9	1,53	114,7	-5,42	118,5	113,6
IFP	1,37	153,8	1,77	138,8	-37,50	155,3	136,4
IFD	-	-	-	-	-	179,1	174,6

Tabla 4-14. Ángulos máximos y mínimos y definición de la pendiente de cierre en las cuatro articulaciones del dedo pulgar en agarre en pinza.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
CMC	1,39	152,8	1,87	147,0	-12,08	154,1	146,8
MCF	1,42	160,5	1,74	156,2	-13,44	161,4	154,3
IF	1,19	156,1	1,57	163,5	19,47	165,0	155,7
OP	0,95	147,7	1,53	142,2	-9,48	148,5	141,0

Tabla 4-15. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo anular en mano abierta.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	1,28	149,6	1,67	176,9	70,00	169,2	146,8
IFP	-	-	-	-	-	149,2	138,6
IFD	-	-	-	-	-	178,8	172,3

Tabla 4-16. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo corazón en mano abierta.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	1,23	141,1	1,6	164,9	64,32	170,5	139,2
IFP	-	-	-	-	-	161,1	157,7
IFD	-	-	-	-	-	177,9	168,2

Tabla 4-17. Ángulos máximos y mínimos y definición de la pendiente de cierre en las tres articulaciones del dedo índice en mano abierta.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
MCF	1,41	149,6	1,8	143,3	-16,15	155,3	140,1
IFP	-	-	-	-	-	158,7	153,4
IFD	-	-	-	-	-	179,4	173,2

Tabla 4-18. Ángulos máximos y mínimos y definición de la pendiente de cierre en las cuatro articulaciones del dedo pulgar en mano abierta.

Articulación	X1[s]	Y1[°]	X2[s]	Y2[°]	Pendiente	Máx[°]	Mín[°]
CMC	1,35	161,7	1,93	139,1	-38,97	165,0	135,9
MCF	1,33	176,0	1,79	169,9	-13,26	176,9	167,3
IF	1,55	166,9	2,05	158,9	-16,00	167,6	154,4
OP	1,20	141,0	1,58	147,0	15,79	150,3	139,9

En general, los movimientos realizados presentan una evolución de los ángulos articulares partiendo desde una apertura que se mantiene constante hasta el momento en que se hace contacto con el objeto; luego sigue una fase de aumento o disminución del ángulo que puede ser descrita por medio de una recta con una pendiente determinada. Después del cierre el movimiento finaliza con una fase angular constante que dura hasta la finalización de la grabación.

Para los agarres esférico y cilíndrico se observa que la mayoría de las articulaciones presentan las tres fases de evolución angular mencionadas anteriormente, salvo la articulación metacarpo-falángica del pulgar. Dicha articulación presentó un delta angular de $1,9^\circ$ para el agarre cilíndrico y $5,1^\circ$ para el agarre esférico; rangos mucho menores que los vistos en las demás articulaciones. Estos cambios pequeños indican que la articulación mencionada no genera cambios significativos en los dos tipos de prensión.

Las mediciones realizadas para el agarre plano muestran cuantitativamente que la articulación más influyente para los dedos 2, 3 y 4 es la articulación MCF; lo cual puede hacerse evidente por simple observación. En el dedo 1 es posible apreciar que el ángulo más determinante en el agarre plano es el de oposición.

En el agarre plano se evidencia también que las articulaciones interfalángicas distales para los dedos 2, 3 y 4, y la articulación interfalángica para el dedo 1, son las menos influyentes y las que presentan menor cambio.

Se podría pensar que las variaciones angulares en los agarres de pinza y plano en los dedos índice y pulgar deberían ser muy similares pero esto no se ve reflejado en los datos obtenidos pues las pendientes de cambio y los ángulos máximos en las articulaciones difieren significativamente; sin embargo, los ángulos mínimos son muy cercanos. Esto puede deberse a que las posturas al inicio de las tomas para cada movimiento fueron distintas, es decir, un movimiento inició con una apertura mayor que la del otro. A partir de esta observación surge la necesidad de estandarizar las posturas iniciales y el tiempo de agarre en pruebas futuras.

La postura de mano abierta muestra una variación angular superior en la articulación MCF para el 2do, 3er y 4to dedo y en la articulación CMC para el 1er dedo. Esta variación indica que el cambio desde una posición de relajación hacia una postura de mano abierta no representa mayor amplitud de movimiento para el resto de las articulaciones.

4.2 Captura Final

Como se mencionó en el Capítulo 2, fueron ubicados diferentes marcadores en la mano para realizar las capturas de movimiento. En total se emplearon once marcadores distribuidos en el dorso de la mano, tal y como se muestra en la Figura 4-2. Para la realización de las tomas para el agarre en pinza se retiraron los marcadores ubicados en los dedos 3 y 4. Durante las tomas de agarre esférico, cada individuo posaba su mano en posición de relajación sobre una esfera de poliestireno expandido de aproximadamente 49 mm de diámetro, y posteriormente se realizaba un agarre completo de dicho objeto. Durante las tomas de agarre cilíndrico, cada individuo posaba su mano en posición de relajación sobre un cilindro de poliestireno expandido de aproximadamente 30 mm de diámetro, y posteriormente se realizaba un agarre completo de dicho objeto. Para los agarres pinza (bidigital) y plano cada individuo ejecutó el agarre de una tabla pequeña de balsa de 4 mm de espesor.

Por cada toma fueron calculados cinco ángulos en la mano distribuidos así: articulación interfalángica (IF) del dedo 1 (pulgar), articulación de oposición (OP) y articulaciones metacarpo-falángicas (MCF) del dedo 2 (índice), 3 (corazón) y 4 (anular). De cada evolución angular se identificaron el ángulo máximo y mínimo, el tiempo de inicio y fin de la fase de cierre, la pendiente promedio del cierre, la pendiente máxima del cierre y la pendiente ajustada (Figura 4-3); lo anterior arrojó un total de 3780 datos extraídos. En las Tabla 4-19 a Tabla 4-22 se resumen los resultados arrojados por la captura de movimiento, mostrando en las columnas de izquierda a derecha los promedios de deltas angulares ($\Delta\theta = \theta_{\max} - \theta_{\min}$), desviaciones estándar de deltas angulares, promedios de deltas de tiempo ($\Delta t = \text{tiempo final del cierre} - \text{tiempo inicial del cierre}$), desviaciones estándar de los deltas de tiempo, promedio de pendientes ajustadas ($\Delta\theta/\Delta t$), desviaciones estándar de las pendientes ajustadas, coeficientes de variación de las pendientes ajustadas, el promedio de tiempo de inicio (momento en el cual se inicia el cierre de la articulación) y por último el orden de cierre (1° a 5°) de las articulaciones con respecto al tiempo de inicio respectivamente.



Figura 4-2. Marcadores posicionados en el dorso de la mano.

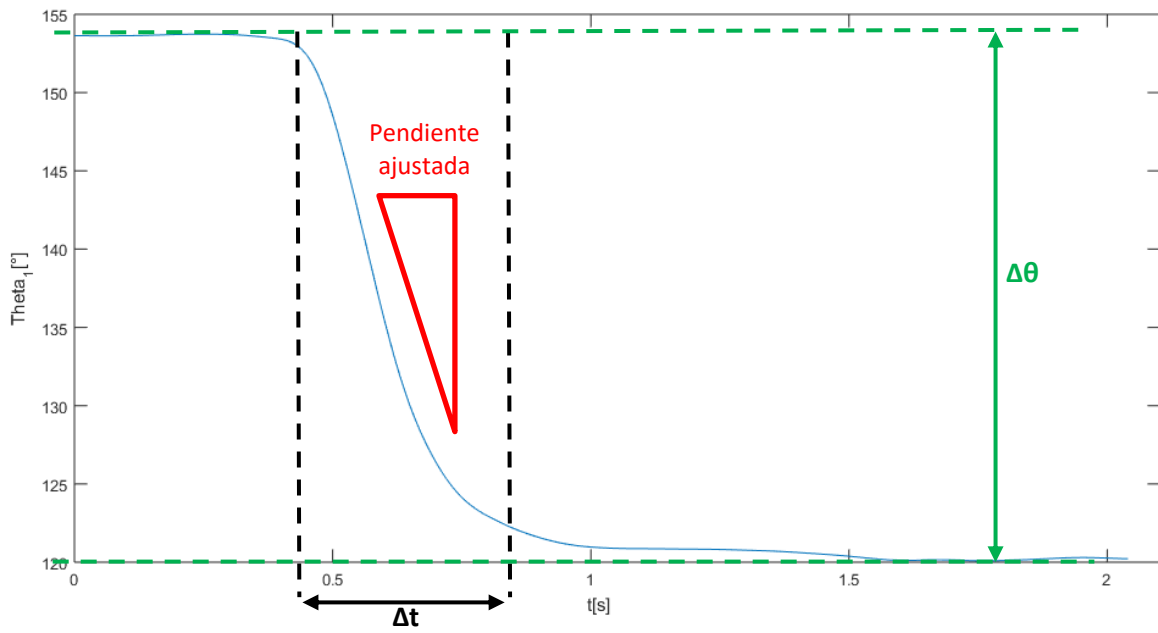


Figura 4-3. Ilustración de variables extraídas por cada articulación.

Tabla 4-19. Datos arrojados para el agarre cilíndrico.

Articulación	θ Máx Prom [°]	θ Mín Prom [°]	$\Delta\theta$ Prom [°]	$\Delta\theta$ DesvEst [°]	Δt Prom [s]	Δt DesvEst [s]	PA Prom [°/s]	PA DesvEst [°/s]	Coef Var PA	Prom T de Inicio [s]	Orden
Dedo 1	172,65	130,69	41,960	9,4624	0,6413	0,2640	76,186	32,069	0,420	0,513	5
OP	135,26	117,93	17,324	7,3350	0,8636	0,3718	24,952	15,424	0,618	0,081	1
Dedo 2	151,85	116,90	34,954	6,1878	0,6646	0,2219	57,460	19,929	0,346	0,117	4
Dedo 3	145,43	104,60	40,826	13,605	0,6096	0,2591	74,900	32,236	0,430	0,074	2
Dedo 4	163,03	116,74	46,289	12,750	0,5526	0,2704	99,356	48,587	0,489	0,067	3

Tabla 4-20. Datos arrojados para el agarre esférico.

Articulación	θ Máx Prom [°]	θ Mín Prom [°]	$\Delta\theta$ Prom [°]	$\Delta\theta$ DesvEst [°]	Δt Prom [s]	Δt DesvEst [s]	PA Prom [°/s]	PA DesvEst [°/s]	Coef Var PA	Prom T de Inicio [s]	Orden
Dedo 1	170,95	121,88	49,072	10,862	0,7000	0,3089	84,204	38,213	0,453	0,493	5
OP	135,90	115,60	20,298	4,5489	1,1863	0,3384	18,510	6,9156	0,373	0,029	1
Dedo 2	155,46	123,84	31,624	8,8946	0,6803	0,3189	56,349	30,252	0,536	0,119	4
Dedo 3	146,50	118,79	27,717	11,892	0,7813	0,4311	44,730	29,964	0,669	0,109	3
Dedo 4	159,96	131,49	28,464	11,014	0,6303	0,3542	54,866	29,302	0,534	0,083	2

Tabla 4-21. Datos arrojados para el agarre plano.

Articulación	θ Máx Prom [°]	θ Mín Prom [°]	$\Delta\theta$ Prom [°]	$\Delta\theta$ DesvEst [°]	Δt Prom [s]	Δt DesvEst [s]	PA Prom [°/s]	PA DesvEst [°/s]	Coef Var PA	Prom T de Inicio [s]	Orden
Dedo 1	170,28	152,08	18,198	7,0716	0,6453	0,2533	32,504	17,148	0,527	0,199	5
OP	123,26	105,59	17,662	4,8690	0,8336	0,2650	23,213	9,5045	0,409	0,065	1
Dedo 2	153,05	129,67	23,386	11,335	0,7233	0,2440	37,131	25,080	0,675	0,230	2
Dedo 3	143,98	118,33	25,653	9,5170	0,8070	0,2361	35,524	18,664	0,525	0,220	3
Dedo 4	159,04	134,71	24,329	14,208	0,7313	0,2656	34,269	22,213	0,648	0,256	4

Tabla 4-22. Datos arrojados para el agarre en pinza.

Articulación	θ Máx Prom [°]	θ Mín Prom [°]	$\Delta\theta$ Prom [°]	$\Delta\theta$ DesvEst [°]	Δt Prom [s]	Δt DesvEst [s]	PA Prom [°/s]	PA DesvEst [°/s]	Coef Var PA	Prom T de Inicio [s]	Orden
Dedo 1	173,67	157,77	15,901	11,459	0,6236	0,3173	34,309	29,765	0,867	0,210	3
OP	120,20	106,61	13,583	10,669	0,9553	0,2094	14,935	12,867	0,861	0,034	1
Dedo 2	145,37	126,35	19,024	5,259	0,5676	0,1929	36,450	12,240	0,335	0,159	2

Los datos anteriormente presentados sugieren que en promedio, todos los movimientos de agarre analizados inician por el cierre de la articulación de oposición del pulgar y finalizan por la articulación interfalángica de este mismo dedo; mientras que el cierre de los dedos 2, 3 y 4 se ejecuta prácticamente de manera simultánea (salvo para el agarre en pinza durante el cual los dedos 3 y 4 se encuentran en máxima flexión).

Al analizar las pendientes de cierre de los dedos, se hace evidente la variabilidad de esta medida a lo largo de todas las pruebas en todos los tipos de agarre, presentando desviaciones estándar que van desde el 45% y que llegan hasta el 86% de las pendientes promedio.

Existe una gran similitud entre los ángulos máximos de los agarres cilíndrico y esférico, pero también se notan diferencias en los ángulos mínimos de estos agarres. Esto se debe a que el diámetro de la esfera empleada (49 mm) fue mayor que diámetro del cilindro empleado (30 mm), lo cual afecta los ángulos de los dedos 2 a 4 (son mayores dichos valores en las tomas de agarre esférico). Los ángulos OP y del dedo 1 son mayores en el agarre cilíndrico dado que el cilindro manipulado pasaba frente a la palma de la mano, extendiéndose más allá del pliegue entre los dedos 1 y 2.

Entre el agarre plano y el agarre en pinza, comparando únicamente los dedos 1 y 2 y la articulación OP, se observan grandes similitudes tanto en los máximos y mínimos angulares como en las pendientes promedio. Los deltas angulares son similares debido que para ambos agarres se utilizó la misma lámina de balsa como objeto a manipular. Al analizar las pendientes, a pesar de su variabilidad, se puede concluir que la manera en que los dedos se cierran en ambos agarres es muy similar, teniendo en cuenta que durante el agarre en pinza se mantienen flexionados los dedos 3 a 5.

Para todos los tipos de agarre se presentan ángulos máximos muy cercanos, los cuales pueden ser empleados como referente para establecer los valores de una postura de mano abierta en relajación. Los datos también sugieren que todos los agarres, en promedio, presentan duraciones cercanas a 1 segundo.

4.3 Control

Los resultados de la etapa de control se muestran en dos partes. La primera parte corresponde al control en lazo abierto y lo que tiene que ver con adecuaciones electrónicas del prototipo. La segunda parte muestra los resultados de la etapa de control en lazo cerrado, partiendo desde la implementación de la red neuronal y terminando con la definición angular de las posturas a simular por el prototipo.

Interfaz control en lazo abierto

Una vez se selecciona el botón conectar y se ha establecido la comunicación, el usuario puede controlar el movimiento de cada dedo por separado utilizando su respectivo *slider*. En la Figura 4-4 se puede observar un ejemplo de los ángulos definidos en la interfaz, los ángulos medidos por los sensores y su resultado en el prototipo.



Figura 4-4. Valores de los *sliders*, valores de los sensores y posición de la mano de acuerdo a éstos.

Interfaz control en lazo cerrado

La interfaz mostrada en la Figura 3-14 cuenta con cuatro botones. El botón de grabado (“Entrenar”) se encarga de recibir la información a través de un micrófono incorporado o conectado al PC, pidiendo al usuario que repita diez veces cada palabra después de presionar la tecla “Enter”. En la parte inferior de la interfaz se indican las instrucciones a seguir, dentro de un cuadro de texto. Una vez se termine el proceso de grabación se comparan las frecuencias obtenidas para calcular el error relativo.



Figura 4-5. Funcionamiento del botón "Entrenar".

Luego de entrenar la Red Neuronal, el usuario debe esperar el resultado del error relativo, el cual se espera sea menor a 0,01 para evitar posibles confusiones al momento del reconocimiento [42]. En la Tabla 4-23 se muestran tres ejemplos del error calculado.

Tabla 4-23. Ejemplos de cálculo del error relativo de la RNA.

Prueba	Cálculo del error	Tipo de ambiente	Resultado
1	0,0039361	Controlado	Deseado
2	0,0048372	Controlado	Deseado
3	2,0081000	Expuesto	Repetir

Una vez la red ha sido entrenada, y el error es el deseado, se procede a oprimir el botón “Escuchar” (Figura 4-6). Para evitar confusiones y detecciones indeseadas, se definió que antes

de indicar a la mano una acción a ejecutar se tendrá que pronunciar el comando de seguridad "Promanu". El usuario deberá mencionar el comando de seguridad seguido por la acción que desea realice el prototipo. Una vez la acción sea identificada se mostrará una imagen con la acción a realizar.

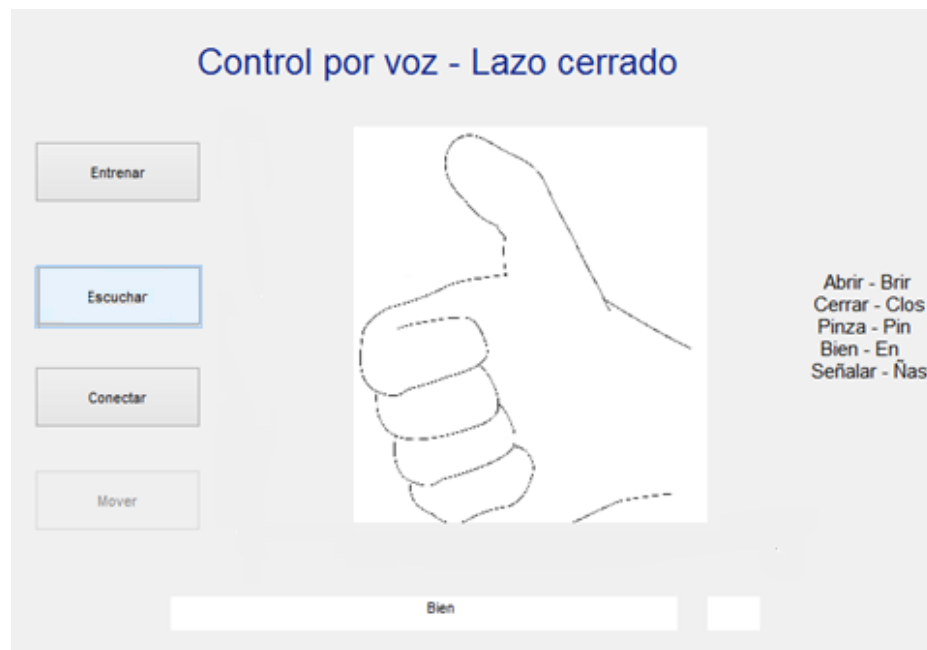


Figura 4-6. Funcionamiento del botón "Escuchar".

Luego de que se haya identificado satisfactoriamente el comando, el botón "Conectar" se encargará de establecer la conexión Matlab® –Arduino®. Una vez está establecida dicha conexión, se prosigue al último paso, el botón "Mover" (se mantiene inactivo mientras no exista conexión con el prototipo), que permite que los motores se accionen para que el prototipo realice la acción deseada. Un ejemplo del prototipo reproduciendo la acción indicada en la Figura 4-6 se muestra en la Figura 4-7.

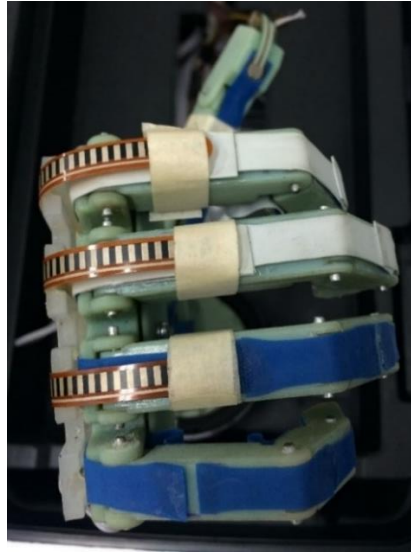


Figura 4-7. Prototipo de prótesis realizando la acción "Bien".

Criterio de Parada:

El criterio de parada para cada articulación consiste en calcular la desviación estándar en una ventana dinámica de los últimos 10 datos medidos por el respectivo sensor de flexión. Cada grado de libertad cuenta con un parámetro de tolerancia diferente ya que los dispositivos de medición, a pesar de ser de la misma referencia y fabricante, son diferentes y se encuentran en cadenas cinemáticas diferentes.

Para calcular el error de funcionamiento de este sistema se realizaron dos tipos de pruebas. La primera fue una sesión de 30 accionamientos de flexión y extensión de los dedos sin objetos que obstaculizaran el movimiento, tomando como error la detención tardía o temprana de la articulación. La segunda sesión también contó con 30 repeticiones de flexión de los dedos, pero esta vez agarrando un objeto, tomando como error los mismos criterios de la primera prueba. Los datos de estas pruebas se encuentran en la Tabla 4-24.

Tabla 4-24. Porcentaje de éxito del criterio de parada.

Prueba	Dedo 1	OP	Dedo 2	Dedo 3	Dedos 4 y 5
Libre	90,0%	96,7%	93,3%	93,3%	96,7%
Objeto	97,0%	90,0%	96,7%	90,0%	90,0%

Como se puede observar en la tabla anterior, el criterio de parada presenta errores entre 3,3% y 10%, lo cual puede considerarse alto para este tipo de accionamientos. Sin embargo, durante el desarrollo de esta investigación y particularmente durante las pruebas del criterio de parada, se observaron diferentes falencias en el sistema mecánico del prototipo que en la mayoría de los casos influyen en el error del sistema de parada.

Los sistemas de accionamiento por guayas, que son los que conforman la totalidad de los accionamientos de los dedos, son excesivamente susceptibles a daños por rupturas en las guayas o desenchajamiento de las mismas en las poleas. Estas rupturas y desacoples con las poleas se deben que las trayectorias que deben recorrer los cables de tensión desde el motor hasta el extremo de cada dedo son extensas en algunos casos y en otros casos presentan obstáculos como cables de energía, motores, otras guayas o cambios de curvatura que aumentan la fricción y la tensión, incrementando el riesgo de ruptura.

Es por lo anterior que para trabajos futuros se recomienda diseñar mejores guías de guayas y ubicación más eficiente de motores o preferiblemente utilizar sistemas de accionamiento diferentes como por ejemplo mecanismos de barras actuadas con servomotores.

|

5. Conclusiones y Recomendaciones

5.1 Conclusiones

Del análisis de los resultados de la captura del movimiento de la mano se puede concluir, que en general todos los movimientos de agarre analizados inician con el cierre de la articulación de oposición del pulgar y finalizan por la articulación interfalángica de este mismo dedo; mientras que el cierre de los dedos 2, 3 y 4 se ejecuta prácticamente de manera simultánea (salvo para el agarre en pinza durante el cual los dedos 3 y 4 se encuentran en máxima flexión).

Existe una gran similitud entre los ángulos máximos de los agarres cilíndrico y esférico, pero también se notan diferencias en los ángulos mínimos de estos agarres debido a que el diámetro de la esfera empleada fue mayor que diámetro del cilindro utilizado. Esto sugiere para la prótesis la posibilidad de emplear una única secuencia de cierre tanto para objetos cilíndricos como para esféricos.

Los datos indican que para el agarre plano, las articulaciones que mayor variación presentan son las del dedo 1 y la articulación OP, mientras que las articulaciones MCF de los dedos 2, 3 y 4 presentan cambios pequeños. Durante las tomas de captura de movimiento se evidenció que un agarre plano se logra cerrando las articulaciones MCF y dejando en extensión las articulaciones interfalángicas de los dedos; esto no es realizable en el prototipo de prótesis dado que los accionamientos de las articulaciones de los dedos son dependientes.

Dada la variabilidad indicada por los coeficientes de variación mostrados, entre la Tabla 4-19 y la Tabla 4-22, de las velocidades de cierre (pendientes) determinadas para las articulaciones de los dedos por cada tipo de agarre, no resultó conveniente obtener expresiones para la descripción cinemática de la mano. Sin embargo, la información extraída permite definir

posturas iniciales, secuencias de activación de los cierres de los dedos y valores promedios para la razón de cambio angular de cada articulación por cada tipo de prensión.

Para todos los tipos de agarre se presentan ángulos máximos muy cercanos, los cuales pueden ser empleados como referente para establecer los valores de una postura de mano abierta en relajación. Los datos también indican que todos los agarres ocurren con duraciones cercanas en promedio a 1 segundo.

La detección de comandos de voz obedece únicamente a un usuario (quien realizó el entrenamiento de la red neuronal) debido a que los pesos sinápticos de las neuronas varían según el tono, frecuencia y timbre de la voz del usuario.

Se apreció la necesidad de implementar un comando de seguridad con un alto nivel de precisión debido a que la red reconocía erróneamente los comandos. Por ejemplo, al mencionar cualquier palabra diferente a los comandos predeterminados, la red neuronal la procesaba y la clasificaba dentro del comando con el que tuviera mayor similitud.

Al aumentar los niveles de exigencia de la red como la cantidad de repeticiones por comando, el número de coeficientes LPC y las iteraciones del cálculo del error, la detección de comandos se hace más precisa. Por otro lado, la fase de entrenamiento de la red neuronal debe ejecutarse en un ambiente controlado con muy poco ruido ambiental, evitando así errores e interferencias en la señal.

Si bien existen diferentes estrategias de control robusto, como por ejemplo control PID, el tipo de control propuesto (proporcional con histéresis) se plantea como una opción de bajo costo computacional, que puede facilitar posteriormente el traslado de los algoritmos de accionamiento desde el PC a un sistema embebido como Arduino®, Raspberry PI® u otros, disminuyendo así el costo final del producto comercial.

5.2 Recomendaciones

Con el fin de apoyar diseños de prótesis de mano más completos y cercanos a la mano humana, aprovechando las tecnologías actuales de procesamiento de imágenes y captura de movimiento, se recomienda llevar a cabo estudios cinemáticos de la mano que incluyan más articulaciones y puntos de referencia (como por ejemplo las articulaciones inferfalángicas proximales y distales de los dedos).

Buscando un mejor comportamiento de los actuadores del prototipo, se puede implementar un sistema de control electrónico más robusto, por ejemplo PID, reemplazando los micromotores DC por microservos o motores DC de mayor voltaje. Además, desde el punto de vista del diseño mecánico, se recomienda replantear el accionamiento por cables o guayas ya que este tipo de mecanismo presenta baja confiabilidad (son frecuentes las rupturas de las guayas o desencaje de las poleas); adicionalmente un cable por dedo no permite controlar de manera independiente las distintas articulaciones de un dedo.

El dedo pulgar del prototipo utilizado cuenta con dos grados de libertad, mientras que el dedo pulgar humano cuenta con tres. Lo anterior conduce a la necesidad de evaluar el diseño de pulgares de más grados de libertad para las prótesis de mano.

Aprovechando los sistemas de desarrollo actuales como Arduino® y sus diferentes módulos disponibles en el mercado, o las placas Raspberry Pi®, se aconseja implementar sistemas de control independientes de un PC, acercando el prototipo de prótesis ProManu a un producto más fácilmente comercializable.

A. Anexo: Adecuaciones al Prototipo de Prótesis Mano

Durante la etapa de ensamblaje y adecuación del prototipo de prótesis de mano fue necesario reemplazar, adecuar y construir algunas piezas con el fin de asegurar el buen funcionamiento electrónico y mecánico del dispositivo. Inicialmente, debido a averías en los dedos dos y tres, fue necesario reemplazar los ejes plásticos originales (hechos a medida) de las articulaciones MCF por ejes metálicos cilíndricos. Con respecto al diseño original, se cambió el micromotor DC que daba movimiento a la articulación OP por uno de mayor tamaño, reubicando el motor de flexo-extensión del dedo 1 cerca a la articulación de la muñeca; este cambio se da con el objetivo de generar mayor fuerza en dicha articulación. Con el fin de producir la apertura de los dedos se eligieron trozos de bandas elásticas de terapia física (Thera-Band™), adheridos por las caras palmar (para ayudar la flexión) y dorsal (para generar extensión) de cada uno de los dedos. La Figura A. 1 ilustra los mecanismos utilizados para el movimiento de extremidades mencionadas. Para la articulación de rotación del antebrazo se fabricó un nuevo engranaje (Figura A. 2) para controlar el movimiento con un microservo y no con el micromotor DC que se consideró en el diseño original. Esta modificación se realiza con el fin de facilitar el posterior control de este grado de libertad. También fue necesario fabricar una nueva polea por medio de impresión 3D para envolver la guaya de flexo-extensión del pulgar.

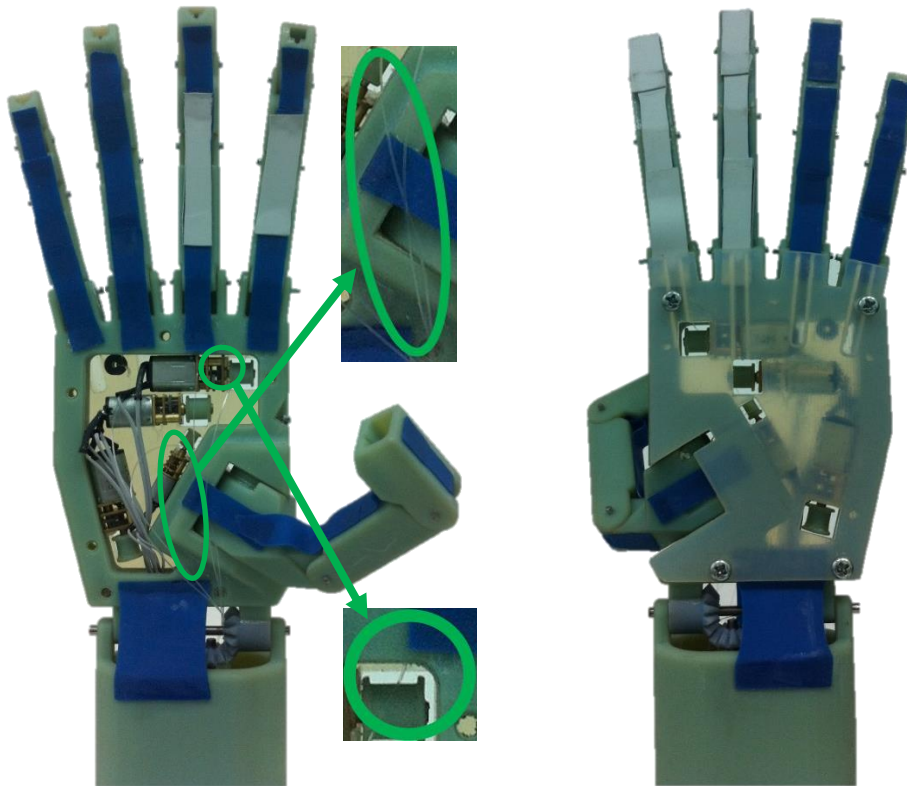


Figura A. 1. Mecanismos de accionamiento de la mano. En los círculos verdes se muestran las guayas y las poleas que las envuelven.

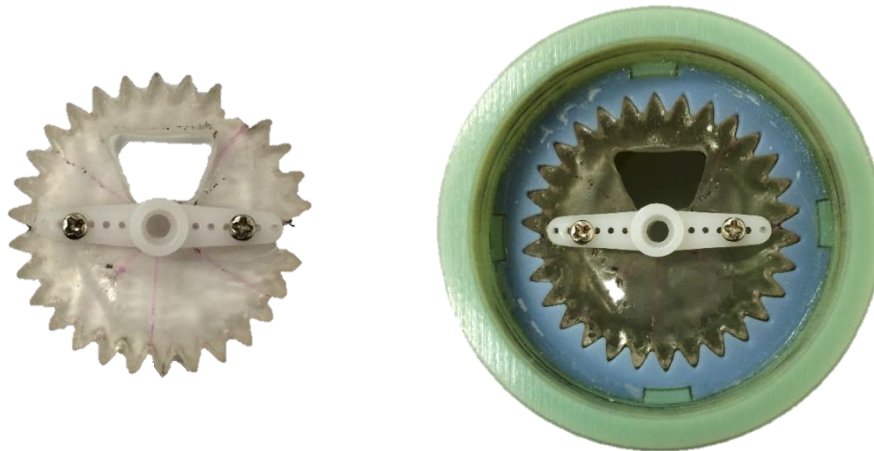


Figura A. 2. Engranaje fabricado para accionar la rotación de la muñeca con el microservo.

Componentes Electrónicos

El dispositivo anteriormente mencionado es accionado por cinco micromotores DC con reducción, con un voltaje de funcionamiento nominal de 5V para los movimientos de los dedos [26], y también posee dos microservos de 5V para los movimientos de flexión y extensión de la muñeca y para la rotación del antebrazo. En la Figura A. 3 se muestra la ubicación de los actuadores. Con el fin de controlar los movimientos de los dedos, se decidió ubicar sensores de flexión resistivos en el dorso de la mano. Para el primer dedo fue necesario utilizar dos sensores dado que éste posee dos grados de libertad (flexo-extensión y OP); para los dedos dos y tres se ubicó un sensor en cada uno y para el control de los dedos cuatro y cinco se posicionó únicamente un sensor en el dedo cuatro ya que estos dos se accionan simultáneamente con el mismo actuador. En la Figura A. 4 se ilustra la ubicación de los sensores.

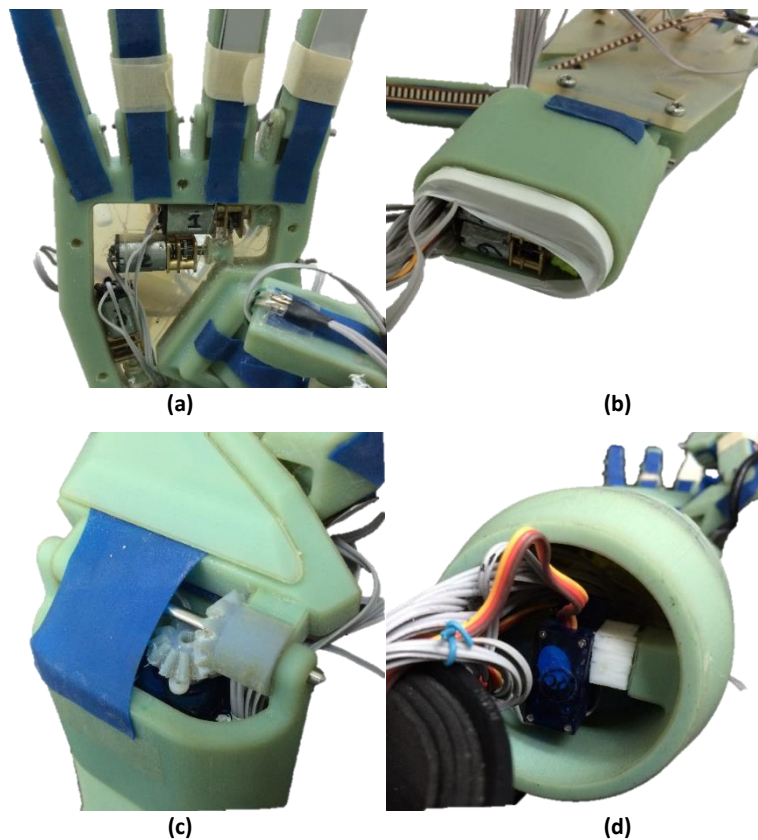


Figura A. 3. Actuadores ubicados en el prototipo.

(a) Motores para la articulación OP y para flexo-extensión de los dedos 2 y 3. (b) Motores para flexo extensión de los dedos 1, 4 y 5. (c) Microservo para flexo-extensión de la muñeca. (d) Microservo para la rotación de la muñeca.



Figura A. 4. Sensores de flexión resistivos para el control de movimiento de los dedos.

Interfaz de potencia

Para proveer de energía suficiente a los actuadores, adecuar y mantener las señales entregadas por los sensores y aislar eléctricamente el dispositivo de control de los demás elementos pasivos del sistema, se diseñó y fabricó una tarjeta de circuito impreso (Figura A. 5) que funciona como interfaz de potencia entre la prótesis de mano y el microcontrolador. Esta tarjeta es alimentada por una fuente de voltaje de computador a 5V y está compuesta por tres circuitos integrados L293D (puente H) para control de motores DC, tres circuitos integrados LM324 (amplificadores operacionales cuádruples) para adecuación de las señales de los sensores, conectores para motores DC y servomotores, y conectores para cinco sensores de flexión y tres sensores adicionales.

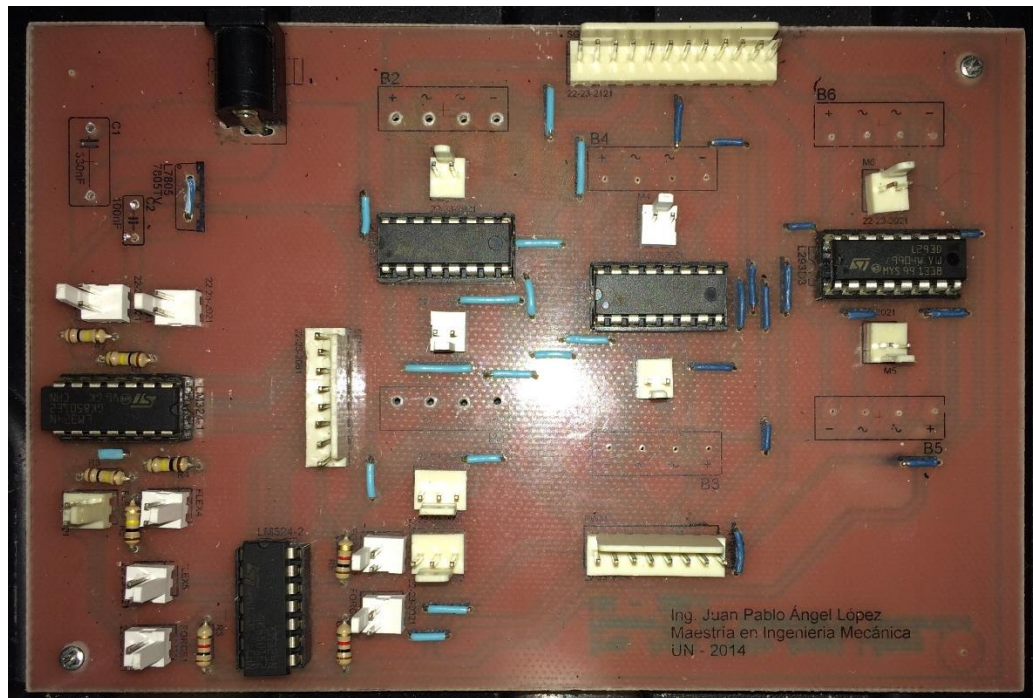


Figura A. 5. Interfaz de potencia para conexión de actuadores y sensores.

Diagrama Esquemático y PCB de la Interfaz de Potencia

A continuación se muestra el diagrama esquemático (Figura A. 6) por componentes pasivos y circuitos integrados utilizados para la fabricación de la interfaz de potencia que permite alimentar los actuadores y acondicionar los sensores del prototipo de prótesis de mano. También se muestra el PCB en sus caras inferior (cobre) (Figura A. 7) y superior (componentes y puentes) (Figura A. 8).

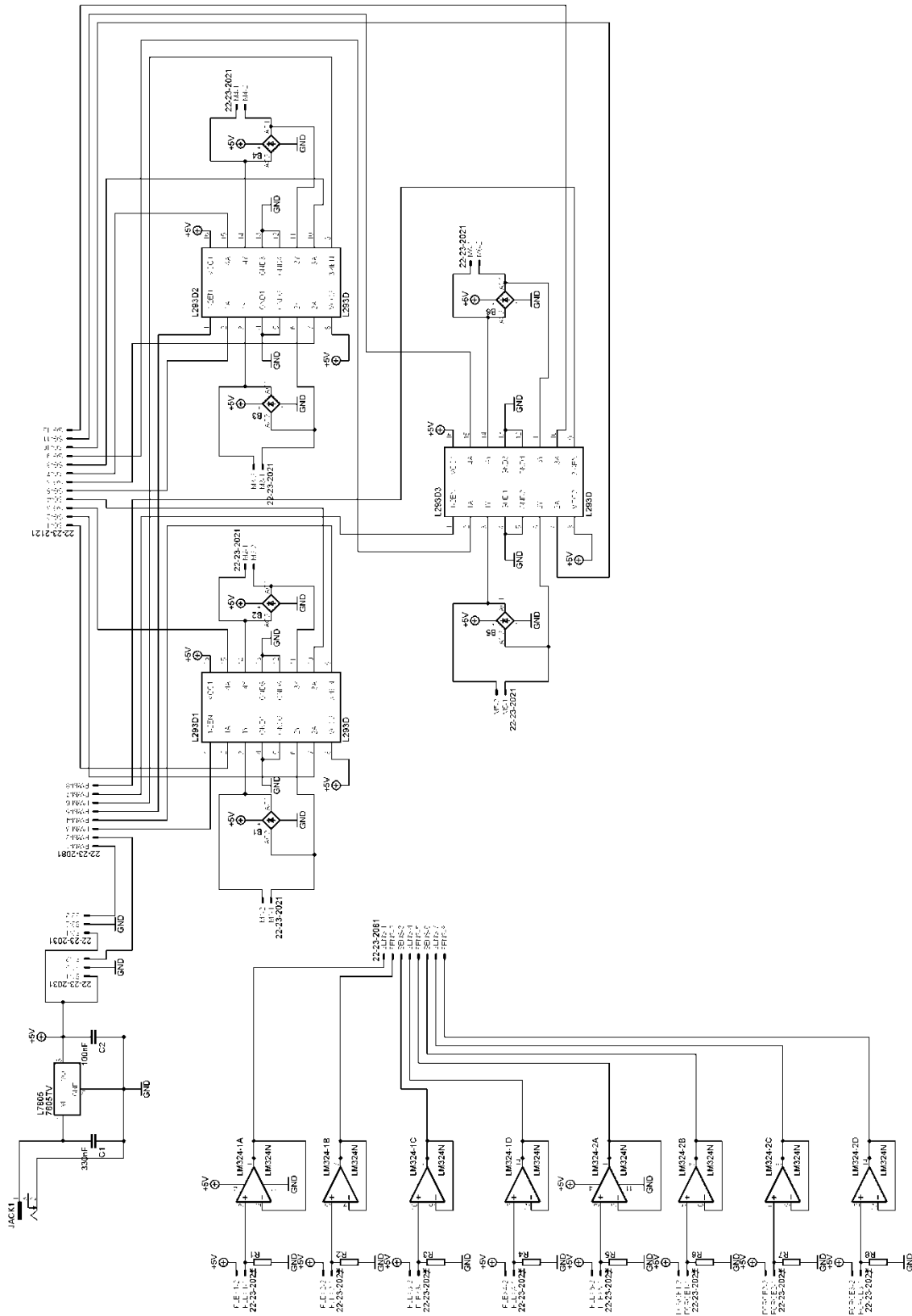


Figura A. 6. Diagrama esquemático de la Interfaz de Potencia.

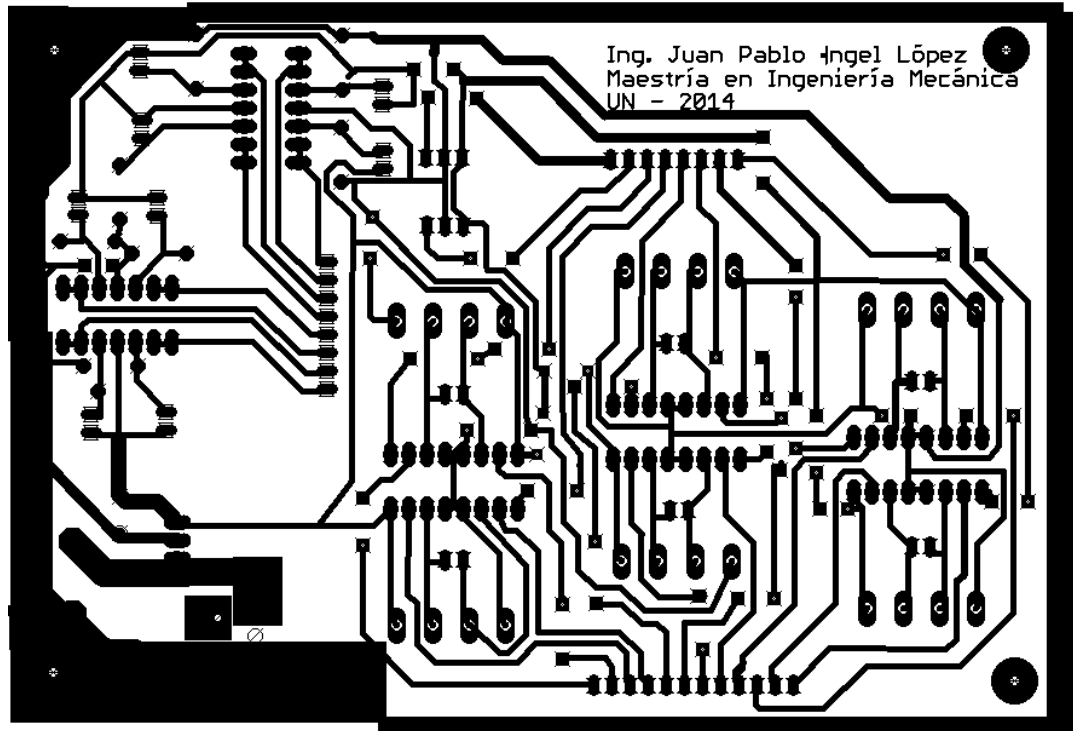


Figura A. 7. Pistas PCB de la Interfaz de Potencia. Vista inferior.

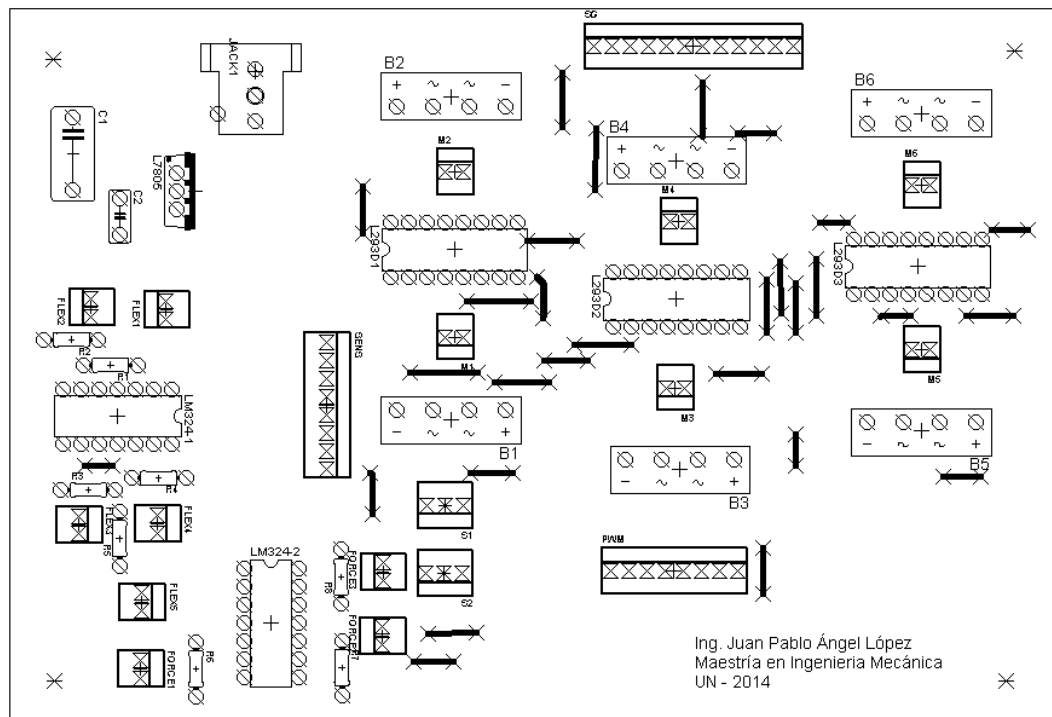


Figura A. 8. Diagrama de ubicación de componentes y puentes para el circuito impreso. Vista superior.

B. Anexo: Códigos escritos en el entorno MATLAB®

- Algoritmo para análisis de datos .c3d

```
clear; close all; clc

prompt = 'Ingrese el nombre del archivo: ';
Nombre = input(prompt, 's');

for mo=1:4
    switch mo
        case 1
            Mov = 'Cilindro';
        case 2
            Mov = 'Esfera';
        case 3
            Mov = 'Pinza';
        case 4
            Mov = 'Plano';
    end

    for r = 1:10
        clearvars -except mo Mov Nombre r; close all; clc
        if r <10
            Archivo = [Nombre, '-', Mov, '_00', num2str(r)];
        else
            Archivo = [Nombre, '-', Mov, '_0', num2str(r)];
        end
        % S = input('Ingrese un valor positivo siendo 1 una velocidad
normal de reproduccion: ');
        S = 1;

        [M, ~] = c3d2mtx(Archivo, []);

        [a, b, ~] = size(M);

        M = sgolayfilt(M, 4, 21);

        plot3dmtx(M(1, :, :), S)
```

```

corr1 = 1;
while corr1~='s' && corr1~='S' %Verificar unidos correctamente
    corr1 = 1;
    corr = 1;
    while corr~='s' && corr~='S' %Verificar vectores ingresados
correctamente
        corr = 1;
        for Dedo = 1:5
            switch Dedo
                case 1
                    d1 = input(['Ingrese los indices de los
marcadores a seguir en el dedo ',num2str(Dedo),': ']);
                    op = input('Ingrese los indices para angulo
de oposicion: ');
                case 2
                    d2 = input(['Ingrese los indices de los
marcadores a seguir en el dedo ',num2str(Dedo),': ']);
                case 3
                    d3 = input(['Ingrese los indices de los
marcadores a seguir en el dedo ',num2str(Dedo),': ']);
                case 4
                    d4 = input(['Ingrese los indices de los
marcadores a seguir en el dedo ',num2str(Dedo),': ']);
                % case 5
                % d5 = input(['Ingrese los indices de los
marcadores a seguir en el dedo ',num2str(Dedo),': ']);
            end
        end
        while corr~='s' && corr~='S' && corr~='n' && corr~='N'
            prompt = '¿Correcto?(S-N): ';
            corr = input(prompt,'s');
        end
    end
end
%Inicializacion de la matriz que contiene los marcadores de interes

%% Construcción de matrices ordenadas para cada dedo

D1 = zeros(a,length(d1),3);
D2 = zeros(a,length(d2),3);
D3 = zeros(a,length(d3),3);
D4 = zeros(a,length(d4),3);
% D5 = zeros(a,length(d5),3);
OP = zeros(a,length(op),3);

for m = 1:length(d1)
    D1(:,m,:) = M(:,d1(m),:); %Dedo 1
end

for m = 1:length(d2)
    D2(:,m,:) = M(:,d2(m),:); %Dedo 2
end

for m = 1:length(d3)
    D3(:,m,:) = M(:,d3(m),:); %Dedo 3

```

```

end

for m = 1:length(d4)
    D4(:,m,:) = M(:,d4(m),:); %Dedo 4
end

% for m = 1:length(d5)
%     D5(:,m,:) = M(:,d5(m),:); %Dedo 5
% end

for m = 1:length(op)
    OP(:,m,:) = M(:,op(m),:); %Marcadores para oposicion
end

%% Grafica de dedos unidos

h = figure(2);
xmin = min(min(M(:,:,1)));
xmax = max(max(M(:,:,1)));
ymin = min(min(M(:,:,2)));
ymax = max(max(M(:,:,2)));
zmin = min(min(M(:,:,3)));
zmax = max(max(M(:,:,3)));

%     prompt = 'Perspectiva (F-T-D-I-S-B-P): ';
%     P = input(prompt,'s');
P = 'S';
v = 1;
%     v = 0;
%     while v<0.1
%         v = input('Ingrese la velocidad (min=0.1 || normal=1):
');
%     end
p = round((70/9)*v + (2/9));

switch P
    case {'D','d'}
        az = -80;
        el = 15;
    case {'I','i'}
        az = 80;
        el = 15;
    case {'F','f'}
        az = -10;
        el = 10;
    case {'T','t'}
        az = 190;
        el = 10;
    case {'S','s'}
        az = -10;
        el = 60;
    case {'B','b'}
        az = -10;

```

```

        e1 = -60;
    otherwise
        az = -37.5;
        e1 = 20;
    end

    for f = 1:p:a
        % WinOnTop1(h); %Para mantener el plot al frente
        (requiere WinOnTop.m)
        % k = get(h,'CurrentCharacter');
        % if k~='p'
        t1 = tic; %Toma registro del tiempo al iniciar el
frame
        plot3(D1(f, :, 1), D1(f, :, 2), D1(f, :, 3), '-o') %Dedo1
        hold on
        plot3(D2(f, :, 1), D2(f, :, 2), D2(f, :, 3), '-o') %Dedo2
        hold on
        plot3(D3(f, :, 1), D3(f, :, 2), D3(f, :, 3), '-o') %Dedo3
        hold on
        plot3(D4(f, :, 1), D4(f, :, 2), D4(f, :, 3), '-o') %Dedo4
        hold on
        % plot3(D5(f, :, 1), D5(f, :, 2), D5(f, :, 3), '-o') %Dedo5
        % hold on
        plot3(OP(f, :, 1), OP(f, :, 2), OP(f, :, 3), '-or') %OpDedo1
        hold off
        axis equal
        axis([xmin xmax ymin ymax zmin zmax])
        grid on
        view([az e1])
        title({'Dedos Unidos'; 'Reproduciendo'})
        xlabel('X')
        ylabel('Y')
        zlabel('Z')

        % if f == 1
        % pause
        % else
        t2 = toc(t1); %toma registro del lapso al final
del frame
        pause(abs(p/100-t2)); %Calcula una pausa para
ajustar la simulacion
        % end
        % else
        % title({'Dedos Unidos'; 'Pausa'})
        % waitforbuttonpress;
        % end
    end
    while corrl~='s' && corrl~='S' && corrl~='n' && corrl~='N'
        prompt = '¿Unidos correctamente?(S-N): ';
        corrl = input(prompt, 's');
    end
end
title({'Dedos Unidos'; 'Fin'})
hgsave(Archivo)

```

```

%% Construccion de matriz de angulos articulares

%angif es una función creada para calcular los ángulos entre
pares de vectores

T(:, :, 1) = angif(D1);
T(:, size(T, 2)+1, 1) = angif(OP); %#ok<*SAGROW>
T(:, 1:size(D2, 2)-2, 2) = angif(D2);
T(:, 1:size(D3, 2)-2, 3) = angif(D3);
T(:, 1:size(D4, 2)-2, 4) = angif(D4);
% T(:, 1:size(D5, 2)-2, 5) = angif(D5);

%% Grafica de angulos articulares en forma de subplot y
almacenamiento en hojas de excel

dlt = [1 2 3]; % Vector para eliminar las hojas 1,2 y 3. Si se
trabaja en office 2013 se debe dejar este vector solo con el valor [1]

T = sgolayfilt(T, 3, 15); % Filtro para graficas de angulos

time = (0:0.01:(a/100)-0.01); % Vector de tiempo para graficar
angulo vs segundos

[~, tt, dd] = size(T);

warning('off', 'MATLAB:xlswrite:AddSheet');

warning('off', 'MATLAB:HandleGraphics:ObsoletedProperty:JavaFrame');

for g = 1:dd
    switch g
        case 1
            figure
            for h=1:tt
                F = T(:, h, g); %Angulos del dedo
                dF = diff(F)/0.01; %Derivada de los angulos
                (pendientes)

                dFM = max(dF); %Maximo de las derivadas
                dFm = min(dF); %Minimo de las derivadas
                if abs(dFM) - abs(dFm) > 0 %Diferencia de max y
min para identificar tipo de cierre
                    mm = dFM;
                else
                    mm = dFm;
                end
                col = tt;
                subplot(1, col, h);
                plot(time, F)
                xlabel('t[s]')
                ylabel(['Theta', '_', num2str(h), ''], ['°'])
                title([Nombre, '-', Mov, '-', num2str(r)], ['Ang
', num2str(h), ' Dedo ', num2str(g)], ...)
            end
        end
    end
end

```

```

                                ['Max:', num2str(max(F)), ' -
', 'Min:', num2str(min(F))], ['mPeak:', num2str(mm) ]})
                                %
                                xlsxwrite([Archivo, '-
Dedo', num2str(g), '.xlsx'], [time' T(:,h,g)], ['Angulo', num2str(h)])
                                end
                                hgsave([Archivo, '-Dedo', num2str(g)])
                                %
                                xls_delete_sheets([Archivo, '-
Dedo', num2str(g), '.xlsx'], dlt)
                                case 2
                                figure
                                for h=1:size(D2,2)-2
                                F = T(:,h,g); %Angulos del dedo
                                dF = diff(F)/0.01; %Derivada de los angulos
                                (pendientes)
                                dFM = max(dF); %Maximo de las derivadas
                                dFm = min(dF); %Minimo de las derivadas
                                if abs(dFM) - abs(dFm)>0 %Diferencia de max y
min para identificar tipo de cierre
                                mm = dFM;
                                else
                                mm = dFm;
                                end
                                col = size(D2,2)-2;
                                subplot(1,col,h);
                                plot(time,F)
                                xlabel('t[s]')
                                ylabel(['Theta', '_{', num2str(h), '}', '°'])
                                title(['Nombre, '- ', Mov, '- ', num2str(r)], ['Ang
', num2str(h), ' Dedo ', num2str(g)], ...
                                ['Max:', num2str(max(F)), ' -
', 'Min:', num2str(min(F))], ['mPeak:', num2str(mm) ]})
                                %
                                xlsxwrite([Archivo, '-
Dedo', num2str(g), '.xlsx'], [time' T(:,h,g)], ['Angulo', num2str(h)])
                                end
                                hgsave([Archivo, '-Dedo', num2str(g)])
                                %
                                xls_delete_sheets([Archivo, '-
Dedo', num2str(g), '.xlsx'], dlt)
                                case 3
                                figure
                                for h=1:size(D3,2)-2
                                F = T(:,h,g); %Angulos del dedo
                                dF = diff(F)/0.01; %Derivada de los angulos
                                (pendientes)
                                dFM = max(dF); %Maximo de las derivadas
                                dFm = min(dF); %Minimo de las derivadas
                                if abs(dFM) - abs(dFm)>0 %Diferencia de max y
min para identificar tipo de cierre
                                mm = dFM;
                                else
                                mm = dFm;
                                end
                                col = size(D3,2)-2;
                                subplot(1,col,h);
                                plot(time,F)
                                xlabel('t[s]')

```



```
        ylabel(['Theta', '_' , num2str(h), ''], '°'])
        title([Nombre, '-', Mov, '-', num2str(r)], ['Ang
', num2str(h), ' Dedo ', num2str(g)], ...
            ['Max:', num2str(max(F)), ' -
', 'Min:', num2str(min(F))], ['mPeak:', num2str(mm)]})
        %
        %       xlswrite([Archivo, '-
Dedo', num2str(g), '.xlsx'], [time' T(:,h,g)], ['Angulo', num2str(h)])
        end
        hgsave([Archivo, '-Dedo', num2str(g)])
        %
        %       xls_delete_sheets([Archivo, '-
Dedo', num2str(g), '.xlsx'], dlt)
        case 4
        figure
        for h=1:size(D4,2)-2
            F = T(:,h,g); %Angulos del dedo
            dF = diff(F)/0.01; %Derivada de los angulos
            (pendientes)
            dFM = max(dF); %Maximo de las derivadas
            dFm = min(dF); %Minimo de las derivadas
            if abs(dFM) - abs(dFm)>0 %Diferencia de max y
min para identificar tipo de cierre
                mm = dFM;
            else
                mm = dFm;
            end
            col = size(D4,2)-2;
            subplot(1,col,h);
            plot(time,F)
            xlabel('t[s]')
            ylabel(['Theta', '_' , num2str(h), ''], '°'])
            title([Nombre, '-', Mov, '-', num2str(r)], ['Ang
', num2str(h), ' Dedo ', num2str(g)], ...
                ['Max:', num2str(max(F)), ' -
', 'Min:', num2str(min(F))], ['mPeak:', num2str(mm)]})
            %
            %       xlswrite([Archivo, '-
Dedo', num2str(g), '.xlsx'], [time' T(:,h,g)], ['Angulo', num2str(h)])
            end
            hgsave([Archivo, '-Dedo', num2str(g)])
            %
            %       xls_delete_sheets([Archivo, '-
Dedo', num2str(g), '.xlsx'], dlt)
            %
            case 5
            %
            %       figure
            %
            %       for h=1:size(D5,2)-2
            %
            %           F = T(:,h,g); %Angulos del dedo
            %
            %           dF = diff(F)/0.01; %Derivada de los angulos
            (pendientes)
            %
            %           dFM = max(dF); %Maximo de las derivadas
            %
            %           dFm = min(dF); %Minimo de las derivadas
            %
            %           if abs(dFM) - abs(dFm)>0 %Diferencia de max y
min para identificar tipo de cierre
                %
                %           mm = dFM;
            %
            %           else
                %
                %           mm = dFm;
            %
            %           end
            %
            %           col = size(D5,2)-2;
```

```

%             subplot(1,col,h);
%             plot(time,F)
%             xlabel('t[s]')
%             ylabel(['Theta','_{',num2str(h),'}','[°]'])
%             title({'Nombre','-',Mov,'-',num2str(r)],[ 'Ang
',num2str(h),' Dedo ',num2str(g)],...
%                 ['Max:',num2str(max(F)),' -
','Min:',num2str(min(F))],[ 'mPeak:',num2str(mm)]})
% %             xlswrite([Archivo,'-
Dedo',num2str(g),'.xlsx'],[time' T(:,h,g)],[ 'Angulo',num2str(h)])
%             end
%             hgsave([Archivo,'-Dedo',num2str(g)])
% %             xls_delete_sheets([Archivo,'-
Dedo',num2str(g),'.xlsx'],dlt)
        end
    end
end
end
end

```

- **Algoritmo para interfaz gráfica de lectura de sensores**

```

function varargout = InterfazLA(varargin)
% INTERFAZLA MATLAB code for InterfazLA.fig
%     INTERFAZLA, by itself, creates a new INTERFAZLA or raises the
existing
%     singleton*.
%
%     H = INTERFAZLA returns the handle to a new INTERFAZLA or the
handle to
%     the existing singleton*.
%
%     INTERFAZLA('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in INTERFAZLA.M with the given input
arguments.
%
%     INTERFAZLA('Property','Value',...) creates a new INTERFAZLA or
raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before InterfazLA_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to InterfazLA_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

```

```

% Edit the above text to modify the response to help InterfazLA

% Last Modified by GUIDE v2.5 20-Jan-2015 10:02:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @InterfazLA_OpeningFcn, ...
                  'gui_OutputFcn',  @InterfazLA_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before InterfazLA is made visible.
function InterfazLA_OpeningFcn(hObject, ~, handles, varargin)
% InterfazLA_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to InterfazLA (see VARARGIN)

% Choose default command line output for InterfazLA
handles.output = hObject;
delete(instrfind)
tic;
clc
global Ve A
A = arduino('COM4');
Ve = 255;

%uiwait(InterfazLA);
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes InterfazLA wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Executes on button press in databutton.
function databutton_Callback(~, ~, handles)
% hObject    handle to databutton (see GCBO)

```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
global i A time
global sens tic
i = 1;
c = 0;
tic; %#ok<VUNUS>
n = 8;
v = 9;

ndatos = v+1;

sens = zeros(ndatos,8);
sen1 = zeros(ndatos,1);
sen2 = zeros(ndatos,1);
sen3 = zeros(ndatos,1);
sen4 = zeros(ndatos,1);
sen5 = zeros(ndatos,1);
sen6 = zeros(ndatos,1);
% sen7 = zeros(ndatos,1);
% sen8 = zeros(ndatos,1);

% m1 = 0.3532;
% c1 = -75.665;
%
% m2 = 0.3580;
% c2 = -149.092;
%
% m3 = 0.3556;
% c3 = -98.4444;
%
% m4 = 0.4138;
% c4 = -163.1034;
%
% m5 = 0.5357;
% c5 = -263.9286;

a1 = 0.0005;
b1 = -0.3631;
c1 = 172.5;

a2 = 0.0014;
b2 = -1.8395;
c2 = 730.34;

a3 = 0.0015;
b3 = -1.7354;
c3 = 620.05;

a4 = 0.0007;
b4 = -0.7797;
c4 = 336.75;

a5 = 0.0015;
```

```

b5 = -1.9322;
c5 = 749.88;

m6 = 1;
c6 = 0;

% m7 = 0;
% c7 = 0;

% m8 = 0;
% c8 = 0;

time = zeros(ndatos,1);

while i == 1
    c = c + 1;
    if c >= v+1
        sen1(c) = A.analogRead(0);
        sen1 = sgolayfilt(sen1,n,v);
        sens(c,1) = round(sen1(c));
        ang1(c) = round((sens(c,1)^2)*a1 + sens(c,1)*b1 + c1);
        %#ok<AGROW>
        ang1 = round(sgolayfilt(ang1,n,v));
        set(handles.flex1,'String',num2str(sens(c,1)))
        set(handles.th1,'String',num2str(ang1(c)))

        sen2(c) = A.analogRead(1);
        sen2 = sgolayfilt(sen2,n,v);
        sens(c,2) = round(sen2(c));
        ang2(c) = round((sens(c,2)^2)*a2 + sens(c,2)*b2 + c2);
        %#ok<AGROW>
        ang2 = round(sgolayfilt(ang2,n,v));
        set(handles.flex2,'String',num2str(sens(c,2)))
        set(handles.th2,'String',num2str(ang2(c)))

        sen3(c) = A.analogRead(2);
        sen3 = sgolayfilt(sen3,n,v);
        sens(c,3) = round(sen3(c));
        ang3(c) = round((sens(c,3)^2)*a3 + sens(c,3)*b3 + c3);
        %#ok<AGROW>
        ang3 = round(sgolayfilt(ang3,n,v));
        set(handles.flex3,'String',num2str(sens(c,3)))
        set(handles.th3,'String',num2str(ang3(c)))

        sen4(c) = A.analogRead(3);
        sen4 = sgolayfilt(sen4,n,v);
        sens(c,4) = round(sen4(c));
        ang4(c) = round((sens(c,4)^2)*a4 + sens(c,4)*b4 + c4);
        %#ok<AGROW>
        ang4 = round(sgolayfilt(ang4,n,v));
        set(handles.flex4,'String',num2str(sens(c,4)))
        set(handles.th4,'String',num2str(ang4(c)))

        sen5(c) = A.analogRead(4);

```

```

sen5 = sgolayfilt(sen5,n,v);
sens(c,5) = round(sen5(c));
ang5(c) = round((sens(c,5)^2)*a5 + sens(c,5)*b5 + c5);
%#ok<AGROW>
ang5 = round(sgolayfilt(ang5,n,v));
set(handles.flex5, 'String', num2str(sens(c,5)))
set(handles.th5, 'String', num2str(ang5(c)))

sen6(c) = A.analogRead(5);
sen6 = sgolayfilt(sen6,n,v);
sens(c,6) = round(sen6(c));
force1(c) = round(sens(c,6)*m6 + c6); %#ok<AGROW>
force1 = round(sgolayfilt(force1,n,v));
set(handles.force1, 'String', num2str(sens(c,6)))
set(handles.fc1, 'String', num2str(force1(c)))

%
% sen7(c) = A.analogRead(6);
% sen7 = sgolayfilt(sen7,n,v);
% sens(c,7) = round(sen7(c));
% force2(c) = round(sens(c,7)*m7 + c7); %#ok<AGROW>
% force2 = round(sgolayfilt(force2,n,v));
% set(handles.force2, 'String', num2str(sens(c,7)))
% set(handles.fc2, 'String', num2str(force2(c)))

%
% sen8(c) = A.analogRead(7);
% sen8 = sgolayfilt(sen8,n,v);
% sens(c,8) = round(sen8(c));
% force3(c) = round(sens(c,8)*m8 + c8); %#ok<AGROW>
% force3 = round(sgolayfilt(force3,n,v));
% set(handles.force3, 'String', num2str(sens(c,8)))
% set(handles.fc3, 'String', num2str(force3(c)))
else
sen1(c) = A.analogRead(0);
sens(c,1) = sen1(c);
ang1 = (sens(c,1)^2)*a1 + sens(c,1)*b1 + c1;
set(handles.flex1, 'String', num2str(sens(c,1)))
set(handles.th1, 'String', num2str(ang1))

sen2(c) = A.analogRead(1);
sens(c,2) = sen2(c);
ang2 = (sens(c,2)^2)*a2 + sens(c,2)*b2 + c2;
set(handles.flex2, 'String', num2str(sens(c,2)))
set(handles.th2, 'String', num2str(ang2))

sen3(c) = A.analogRead(2);
sens(c,3) = sen3(c);
ang3 = (sens(c,3)^2)*a3 + sens(c,3)*b3 + c3;
set(handles.flex3, 'String', num2str(sens(c,3)))
set(handles.th3, 'String', num2str(ang3))

sen4(c) = A.analogRead(3);
sens(c,4) = sen4(c);
ang4 = (sens(c,4)^2)*a4 + sens(c,4)*b4 + c4;
set(handles.flex4, 'String', num2str(sens(c,4)))

```

```

set(handles.th4, 'String', num2str(ang4))

sen5(c) = A.analogRead(4);
sens(c,5) = sen5(c);
ang5 = (sens(c,5)^2)*a5 + sens(c,5)*b5 + c5;
set(handles.flex5, 'String', num2str(sens(c,5)))
set(handles.th5, 'String', num2str(ang5))

sen6(c) = A.analogRead(5);
sens(c,6) = sen6(c);
forcel = round(sens(c,6)*m6 + c6);
set(handles.forcel, 'String', num2str(sens(c,6)))
set(handles.fc1, 'String', num2str(forcel))

%       sen7(c) = A.analogRead(6);
%       sens(c,7) = sen7(c);
%       force2 = round(sens(c,7)*m7 + c7);
%       set(handles.force2, 'String', num2str(sens(c,7)))
%       set(handles.fc2, 'String', num2str(force2))

%       sen8(c) = A.analogRead(7);
%       sens(c,8) = sen8(c);
%       force3 = round(sens(c,8)*m8 + c8);
%       set(handles.force3, 'String', num2str(sens(c,8)))
%       set(handles.fc3, 'String', num2str(force3))
end
time(c) = toc - time(1);
pause(0.001)
end

while i == 1
sens1 = A.analogRead(0);
set(handles.flex1, 'String', num2str(sens1))

sens2 = A.analogRead(1);
set(handles.flex2, 'String', num2str(sens2))

sens3 = A.analogRead(2);
set(handles.flex3, 'String', num2str(sens3))

sens4 = A.analogRead(3);
set(handles.flex4, 'String', num2str(sens4))

sens5 = A.analogRead(4);
set(handles.flex5, 'String', num2str(sens5))

sens6 = A.analogRead(5);
set(handles.forcel, 'String', num2str(sens6))

sens7 = A.analogRead(6);
set(handles.force2, 'String', num2str(sens7))

sens8 = A.analogRead(7);

```

```

        set(handles.force3, 'String', num2str(sens8))
        pause(0.01)
    end

% --- Executes on button press in stopread.
function stopread_Callback(~, ~, ~)
% hObject    handle to stopread (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global i time
global sens

time(1) = 0;
i = 0;

warning('off', 'MATLAB:xlswrite:AddSheet');

for x = 1:5
    xlswrite('Flex.xlsx', [time sens(:,x)], ['Flex ', num2str(x)])
end
k = 0;
for s=x+1:7
    k = k + 1;
    xlswrite('Force.xlsx', [time sens(:,s)], ['Force ', num2str(k)])
end

% --- Outputs from this function are returned to the command line.
function varargout = InterfazLA_OutputFcn(~, ~, handles)
% InterfazLA_OutputFcn(hObject, eventdata, handles)

% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes when selected object is changed in d1.
function d1_SelectionChangeFcn(~, ~, handles)
% d1_SelectionChangeFcn(hObject, eventdata, handles)

% hObject    handle to the selected object in d1
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none
%   was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
global A Ve

```



```
if get(handles.fx1,'Value')
    ad1 = 1;
else
    if get(handles.ex1,'Value')
        ad1= 2;
    else
        ad1 = 0;
    end
end
pines = [34 36 9];
ControlMotor(A,pines,Ve,ad1)

% --- Executes when selected object is changed in d2.
function d2_SelectionChangeFcn(~, ~, handles)
% hObject    handle to the selected object in d2
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none
was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
global A Ve

if get(handles.fx2,'Value')
    ad2 = 1;
else
    if get(handles.ex2,'Value')
        ad2 = 2;
    else
        ad2 = 0;
    end
end
pines = [22 24 6];
ControlMotor(A,pines,Ve,ad2)

% --- Executes when selected object is changed in d3.
function d3_SelectionChangeFcn(~, ~, handles)
% hObject    handle to the selected object in d3
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none
was selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
global A Ve

if get(handles.fx3,'Value')
    ad3 = 1;
else
    if get(handles.ex3,'Value')
        ad3 = 2;
    else
        ad3 = 0;
    end
end
pines = [22 24 6];
ControlMotor(A,pines,Ve,ad3)
```

```

        ad3 = 0;
    end
end
pines = [30 32 8];
ControlMotor(A,pines,Ve,ad3)

% --- Executes when selected object is changed in d45.
function d45_SelectionChangeFcn(~, ~, handles)
% hObject    handle to the selected object in d45
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue:  handle of the previously selected object or empty if none
was selected
%   NewValue:  handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
global A Ve

if get(handles.fx45,'Value')
    ad45 = 1;
else
    if get(handles.ex45,'Value')
        ad45 = 2;
    else
        ad45 = 0;
    end
end
pines = [38 40 10];
ControlMotor(A,pines,Ve,ad45)

% --- Executes when selected object is changed in op.
function op_SelectionChangeFcn(~, ~, handles) %#ok<*DEFNU>
% hObject    handle to the selected object in op
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue:  handle of the previously selected object or empty if none
was selected
%   NewValue:  handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
global A Ve

if get(handles.fxop,'Value')
    aop = 1;
else
    if get(handles.exop,'Value')
        aop = 2;
    else
        aop = 0;
    end
end
pines = [26 28 7];
ControlMotor(A,pines,Ve,aop)

```

```
function ControlMotor(A,pines,vel,ax)
    A.pinMode(pines(1),'output');
    A.pinMode(pines(2),'output');
    switch ax
        case 0
            A.digitalWrite(pines(1),0);
            A.digitalWrite(pines(2),0);
        case 1
            A.digitalWrite(pines(1),0);
            A.digitalWrite(pines(2),1);
        case 2
            A.digitalWrite(pines(1),1);
            A.digitalWrite(pines(2),0);
    end
    A.analogWrite(pines(3),vel);

% --- Executes during object creation, after setting all properties.
function flex1_CreateFcn(~,~,~)
% hObject    handle to flex1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function flex1_DeleteFcn(~,~,~)
% hObject    handle to flex1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function flex2_CreateFcn(~,~,~)
% hObject    handle to flex2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function flex2_DeleteFcn(~,~,~)
% hObject    handle to flex2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function flex3_CreateFcn(~,~,~)
% hObject    handle to flex3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% --- Executes during object deletion, before destroying properties.
function flex3_DeleteFcn(~, ~, ~)
% hObject    handle to flex3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function flex4_CreateFcn(~, ~, ~)
% hObject    handle to flex4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function flex4_DeleteFcn(~, ~, ~)
% hObject    handle to flex4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function flex5_CreateFcn(~, ~, ~)
% hObject    handle to flex5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function flex5_DeleteFcn(~, ~, ~)
% hObject    handle to flex5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function force1_CreateFcn(~, ~, ~)
% hObject    handle to force1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function force1_DeleteFcn(~, ~, ~)
% hObject    handle to force1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes during object creation, after setting all properties.
function force2_CreateFcn(~, ~, ~)
% hObject    handle to force2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function force2_DeleteFcn(~, ~, ~)
% hObject    handle to force2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function force3_CreateFcn(~, ~, ~)
% hObject    handle to force3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function force3_DeleteFcn(~, ~, ~)
% hObject    handle to force3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function th1_CreateFcn(~, ~, ~)
% hObject    handle to th1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function th1_DeleteFcn(~, ~, ~)
% hObject    handle to th1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function th2_CreateFcn(~, ~, ~)
% hObject    handle to th2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% --- Executes during object deletion, before destroying properties.
function th2_DeleteFcn(~, ~, ~)
% hObject    handle to th2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes during object creation, after setting all properties.
function th3_CreateFcn(~, ~, ~)
% hObject    handle to th3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% --- Executes during object deletion, before destroying properties.
function th3_DeleteFcn(~, ~, ~)
% hObject    handle to th3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes during object creation, after setting all properties.
function th4_CreateFcn(~, ~, ~)
% hObject    handle to th4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% --- Executes during object deletion, before destroying properties.
function th4_DeleteFcn(~, ~, ~)
% hObject    handle to th4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes during object creation, after setting all properties.
function th5_CreateFcn(~, ~, ~)
% hObject    handle to th5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% --- Executes during object deletion, before destroying properties.
function th5_DeleteFcn(~, ~, ~)
% hObject    handle to th5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes during object creation, after setting all properties.
function fc1_CreateFcn(~, ~, ~)
% hObject      handle to fc1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function fc1_DeleteFcn(~, ~, ~)
% hObject      handle to fc1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function fc2_CreateFcn(~, ~, ~)
% hObject      handle to fc2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function fc2_DeleteFcn(~, ~, ~)
% hObject      handle to fc2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function fc3_CreateFcn(~, ~, ~)
% hObject      handle to fc3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% --- Executes during object deletion, before destroying properties.
function fc3_DeleteFcn(~, ~, ~)
% hObject      handle to fc3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
```

- Algoritmo para submuestreo de datos y exportación a excel

```

clear all; close all; clc

%% Carga del archivo .c3d
prompt = 'Ingrese el nombre del archivo .c3d: ';
C3D = input(prompt, 's');
M1 = c3d2mtx(C3D, 'n');
M = sgolayfilt(M1, 6, 99);
[f, c, ~] = size(M);

%% Carga del libro de excel
prompt = 'Ingrese el nombre del libro excel: ';
EX1 = input(prompt, 's');
EX = [EX1, '.xlsx'];

%% Animacion de nube de puntos
plot3dmtx(M, 1)

%% Selección de marcadores por articulación
for Dedo = 1:4
    switch Dedo
        case 1
            d1 = input(['Ingrese los indices de los marcadores a seguir
en el dedo ', num2str(Dedo), ': ']);
            op = input('Ingrese los indices para angulo de oposicion:
');
        case 2
            d2 = input(['Ingrese los indices de los marcadores a seguir
en el dedo ', num2str(Dedo), ': ']);
        case 3
            d3 = input(['Ingrese los indices de los marcadores a seguir
en el dedo ', num2str(Dedo), ': ']);
        case 4
            d4 = input(['Ingrese los indices de los marcadores a seguir
en el dedo ', num2str(Dedo), ': ']);
    end
end

%% Construcción de matrices ordenadas por articulación

D1 = zeros(f, length(d1), 3);
D2 = zeros(f, length(d2), 3);
D3 = zeros(f, length(d3), 3);
D4 = zeros(f, length(d4), 3);
OP = zeros(f, length(op), 3);

for m = 1:length(d1)
    D1(:, m, :) = M(:, d1(m), :); %Dedo 1
end

```



```

for m = 1:length(d2)
    D2(:,m,:) = M(:,d2(m),:); %Dedo 2
end

for m = 1:length(d3)
    D3(:,m,:) = M(:,d3(m),:); %Dedo 3
end

for m = 1:length(d4)
    D4(:,m,:) = M(:,d4(m),:); %Dedo 4
end

for m = 1:length(op)
    OP(:,m,:) = M(:,op(m),:); %Marcadores para oposicion
end

%% Gráfica de dedos unidos

figure
xmin = min(min(M(:, :, 1)));
xmax = max(max(M(:, :, 1)));
ymin = min(min(M(:, :, 2)));
ymax = max(max(M(:, :, 2)));
zmin = min(min(M(:, :, 3)));
zmax = max(max(M(:, :, 3)));

plot3(D1(1, :, 1), D1(1, :, 2), D1(1, :, 3), '-o') %Dedo1
hold on
plot3(D2(1, :, 1), D2(1, :, 2), D2(1, :, 3), '-o') %Dedo2
hold on
plot3(D3(1, :, 1), D3(1, :, 2), D3(1, :, 3), '-o') %Dedo3
hold on
plot3(D4(1, :, 1), D4(1, :, 2), D4(1, :, 3), '-o') %Dedo4
hold on
plot3(OP(1, :, 1), OP(1, :, 2), OP(1, :, 3), '-or') %OpDedo1
hold off
axis([xmin xmax ymin ymax zmin zmax])
xlabel(' [m] ')
ylabel(' [m] ')
zlabel(' [m] ')
grid on
title('Dedos unidos')

%% Construcción de matriz de ángulos articulares

%angif es una función creada para calcular los ángulos entre pares de
vectores

T(:, :, 1) = angif(D1);
T(:, :, 2) = angif(OP);
T(:, 1:size(D2, 2)-2, 3) = angif(D2);
T(:, 1:size(D3, 2)-2, 4) = angif(D3);

```

```

T(:,1:size(D4,2)-2,5) = angif(D4);
T = sgolayfilt(T,3,15); % Filtro para gráficas de ángulos

%% Lectura de datos de sensores

S1 = xlsread(EX,'Flex 1');
S(:,1,1) = S1(:,2);
S2 = xlsread(EX,'Flex 2');
S(:,1,2) = S2(:,2);
S3 = xlsread(EX,'Flex 3');
S(:,1,3) = S3(:,2);
S4 = xlsread(EX,'Flex 4');
S(:,1,4) = S4(:,2);
S5 = xlsread(EX,'Flex 5');
S(:,1,5) = S5(:,2);
Ts = S1(:,1);
TS = zeros(length(Ts),1);

for l = 1:length(Ts)
    temp = round(Ts(l)*100);
    TS(l) = temp/100; % Aproximación de tiempo a 1/100 seg
end

%% Muestreo de ángulos calculados

TA = zeros(f,1);
for fr = 1:f
    time = fr-1;
    TA(fr) = time/100; % Vector de tiempo para ángulos calculados
end

AM = zeros(length(TS),1,5);

for d=1:5
    for t2=1:f
        for t1=1:length(TS)
            if TA(t2)==TS(t1)
                AM(t1,1,d)=T(t2,1,d);
            end
        end
    end
end

%% Almacenamiento de variables

warning('off','MATLAB:xlswrite:AddSheet')
for o=1:5
    xlswrite(['Calib-',C3D, '.xlsx'],[TS S(:, :, o)
AM(:, :, o)], ['Theta', num2str(o)])
end
xls_delete_sheets(['Calib-',C3D, '.xlsx'],1);

```

```

%% Gráficas de variables

for g = 1:5
    figure
    subplot(2,1,1), plot(TS,S(:, :,g))
    xlabel('t[s]')
    ylabel('V[bits]')
    title(['Sensor ', num2str(g)])
    subplot(2,1,2), plot(TS,AM(:, :,g))
    xlabel('t[s]')
    ylabel('\theta[°]')
    title(['\theta_', num2str(g)])
    hgsave(['CalibSens', num2str(g), '-', C3D])
end

```

- **Algoritmo para interfaz gráfica de control en lazo abierto**

```

function varargout = Lazoabierto(varargin)
% LAZOABIERTO MATLAB code for Lazoabierto.fig
%     LAZOABIERTO, by itself, creates a new LAZOABIERTO or raises the
existing
%     singleton*.
%
%     H = LAZOABIERTO returns the handle to a new LAZOABIERTO or the
handle to
%     the existing singleton*.
%
%     LAZOABIERTO('CALLBACK',hObject,eventData,handles,...) calls the
local
%     function named CALLBACK in LAZOABIERTO.M with the given input
arguments.
%
%     LAZOABIERTO('Property','Value',...) creates a new LAZOABIERTO or
raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before Lazoabierto_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to Lazoabierto_OpeningFcn via
varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

```

```

% Edit the above text to modify the response to help Lazoabierto

% Last Modified by GUIDE v2.5 13-Jul-2015 17:59:48

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @Lazoabierto_OpeningFcn, ...
                  'gui_OutputFcn',  @Lazoabierto_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
end

% --- Executes just before Lazoabierto is made visible.
function Lazoabierto_OpeningFcn(hObject,~, handles, varargin)
clc
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Lazoabierto (see VARARGIN)

% Choose default command line output for Lazoabierto
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
delete(instrfind)
global vel v f1 f2 f3 f4 f5 h n k ndatos
global pines1 pines2 pines3 pines4 pines5
global sens sen1 sen2 sen3 sen4 sen5
global a1 b1 c1 a2 b2 c2 a3 b3 c3 a4 b4 c4 a5 b5 c5

vel=255;
k = 0;
n = 8;
v = 9;

pines1 = [34 36 9];
pines2 = [26 28 7];
pines3 = [22 24 6];
pines4 = [30 32 8];
pines5 = [38 40 10];

```

```
ndatos=v+1;
sens = zeros(ndatos,8);%sensores
sen1 = zeros(ndatos,1);
sen2 = zeros(ndatos,1);
sen3 = zeros(ndatos,1);
sen4 = zeros(ndatos,1);
sen5 = zeros(ndatos,1);

f1 = 0;
f2 = 0;
f3 = 0;
f4 = 0;
f5 = 0;
h = 5;

a1 = 0.0005;
b1 = -0.3631;
c1 = 172.5;

a2 = 0.0014;
b2 = -1.8395;
c2 = 730.34;

a3 = 0.0015;
b3 = -1.7354;
c3 = 620.05;

a4 = 0.0007;
b4 = -0.7797;
c4 = 336.75;

a5 = 0.0015;
b5 = -1.9322;
c5 = 749.88;

% UIWAIT makes Lazoabierto wait for user response (see UIRESUME)
% uiwait(handles.figure1);

end
% --- Outputs from this function are returned to the command line.

function varargout = Lazoabierto_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
%
```

```

end

% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
%#ok<*INUSL,*DEFNU>
% hObject handle to slider1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
% get(hObject,'Min') and get(hObject,'Max') to determine range
of slider
global A1 n v k sens sen1 vel pines1 A f1 a1 b1 c1 h
A1 = round(get(hObject,'Value')) + 120;
set(handles.text8,'String',num2str(A1));

f1 = 0;

if isempty(A)==0
    while f1 ~= 1
        k = k+1;
        if k >= v+1
            sen1(k) = A.analogRead(0);
            sen1 = sgolayfilt(sen1,n,v);
            sens(k,1) = round(sen1(k));
            angl(k) = round((sens(k,1)^2)*a1 + sens(k,1)*b1 +
c1); %#ok<*AGROW>
            angl = round(sgolayfilt(angl,n,v));
            set(handles.text13,'String',num2str(angl(k)));
            if f1 ~= 1
                if A1 < angl(k)
                    ControlMotor(A,pines1,vel,1);
                    f1 = 0;
                end
                if A1 > angl(k)
                    ControlMotor(A,pines1,vel,2);
                    f1 = 0;
                end
                if (A1 + h > angl(k)) && (A1-h < angl(k))
                    ControlMotor(A,pines1,vel,0);
                    f1 = 1;
                end
            end
        end
    else
        sen1(k) = A.analogRead(0);
        sens(k,2) = sen1(k);
        angl = (sens(k,1)^2)*a1 + sens(k,1)*b1 + c1;
    end
end
end

% --- Executes during object creation, after setting all properties.

```

```

function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
end

% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

global A2 n v k sens sen2 vel pines2 A f2 a2 b2 c2 h
A2 = round(get(hObject,'Value')) + 135;
set(handles.text9,'String',num2str(A2));

f2 = 0;

if isempty(A)==0
    while f2 ~= 1
        k = k+1;
        if k >= v+1
            sen2(k) = A.analogRead(1);
            sen2 = sgolayfilt(sen2,n,v);
            sens(k,1) = round(sen2(k));
            ang2(k) = round((sens(k,2)^2)*a2 + sens(k,2)*b2 + c2);
            ang2 = round(sgolayfilt(ang2,n,v));
            set(handles.text14,'String',num2str(ang2(k)));

            if f2 ~= 1
                if A2 < ang2(k)
                    ControlMotor(A,pines2,vel,1);
                    f2 = 0;
                end
                if A2 > ang2(k)
                    ControlMotor(A,pines2,vel,2);
                    f2 = 0;
                end
                if (A2 + h > ang2(k)) && (A2-h < ang2(k))
                    ControlMotor(A,pines2,vel,0);
                    f2 = 1;
                end
            end
        end
    end
end

```

```

        else
            sen2(k) = A.analogRead(1);
            sens(k,2) = sen2(k);
            ang2 = (sens(k,2)^2)*a2 + sens(k,2)*b2 + c2;
        end
    end
end
end

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
end

% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

global A3 n v k sens sen3 vel pines3 A f3 a3 b3 c3 h
A3 = round(get(hObject,'Value')) + 125;
set(handles.text10,'String',num2str(A3));

f3 = 0;

if isempty(A)==0
    while f3 ~= 1
        k = k+1;
        if k >= v+1
            sen3(k) = A.analogRead(2);
            sen3 = sgolayfilt(sen3,n,v);
            sens(k,1) = round(sen3(k));
            ang3(k) = round((sens(k,3)^2)*a3 + sens(k,3)*b3 + c3);
            ang3 = round(sgolayfilt(ang3,n,v));
            set(handles.text15,'String',num2str(ang3(k)));

            if f3 ~= 1
                if A3 < ang3(k)
                    ControlMotor(A,pines3,vel,1);
                end
            end
        end
    end
end

```



```

        f3 = 0;
    end
    if A3> ang3(k)
        ControlMotor(A,pines3,vel,2);
        f3 = 0;
    end
    if (A3 + h > ang3(k)) && (A3-h < ang3(k))
        ControlMotor(A,pines3,vel,0);
        f3 = 1;
    end
else
    sen3(k) = A.analogRead(2);
    sens(k,3) = sen3(k);
    ang3 = (sens(k,3)^2)*a3 + sens(k,3)*b3 + c3;
end
end
end
end

% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles) %#ok<*INUSD>
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
end

% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

global A4 n v k sens sen4 vel pines4 A f4 a4 b4 c4 h
A4 = round(get(hObject,'Value')) + 145;
set(handles.text11,'String',num2str(A4));

f4 = 0;

if isempty(A)==0
    while f4 ~= 1
        k = k+1;
        if k >= v+1

```

```

sen4(k) = A.analogRead(3);
sen4 = sgolayfilt(sen4,n,v);
sens(k,1) = round(sen4(k));
ang4(k) = round((sens(k,4)^2)*a4 + sens(k,4)*b4 + c4);
ang4 = round(sgolayfilt(ang4,n,v));
set(handles.text16, 'String', num2str(ang4(k)));

if f4 ~= 1
    if A4 < ang4(k)
        ControlMotor(A,pines4,vel,1);
        f4 = 0;
    end
    if A4 > ang4(k)
        ControlMotor(A,pines4,vel,2);
        f4 = 0;
    end
    if (A4 + h > ang4(k)) && (A4-h < ang4(k))
        ControlMotor(A,pines4,vel,0);
        f4 = 1;
    end
else
    sen4(k) = A.analogRead(3);
    sens(k,4) = sen4(k);
    ang4 = (sens(k,4)^2)*a4 + sens(k,4)*b4 + c4;
end
end
end
end

% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
end

% --- Executes on slider movement.
function slider5_Callback(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider

```

```

global A5 n v k sens sen5 vel pines5 A f5 a5 b5 c5 h
A5 = round(get(hObject,'Value')) + 135;
set(handles.text12,'String',num2str(A5));

f5 = 0;

if isempty(A)==0
    while f5 ~= 1
        k = k+1;
        if k >= v+1
            sen5(k) = A.analogRead(4);
            sen5 = sgolayfilt(sen5,n,v);
            sens(k,1) = round(sen5(k));
            ang5(k) = round((sens(k,5)^2)*a5 + sens(k,5)*b5 + c5);
            ang5 = round(sgolayfilt(ang5,n,v));
            set(handles.text17,'String',num2str(ang5(k)));

            if f5 ~= 1
                if A5 < ang5(k)
                    ControlMotor(A,pines5,vel,1);
                    f5 = 0;
                end
                if A5 > ang5(k)
                    ControlMotor(A,pines5,vel,2);
                    f5 = 0;
                end
                if (A5 + h > ang5(k)) && (A5-h < ang5(k))
                    ControlMotor(A,pines5,vel,0);
                    f5 = 1;
                end
            end
        else
            sen5(k) = A.analogRead(4);
            sens(k,1) = sen5(k);
            ang5 = (sens(k,5)^2)*a5 + sens(k,5)*b5 + c5;
        end
    end
end
end
end
% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.

end

function ControlMotor(A,pines,vel,ax)
    A.pinMode(pines(1),'output');
    A.pinMode(pines(2),'output');

```

```

switch ax
    case 0
        A.digitalWrite(pines(1),0);
        A.digitalWrite(pines(2),0);
    case 1
        A.digitalWrite(pines(1),0);
        A.digitalWrite(pines(2),1);
    case 2
        A.digitalWrite(pines(1),1);
        A.digitalWrite(pines(2),0);
end
A.analogWrite(pines(3),vel);
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global A v n sens ang1 ang2 ang3 ang4 ang5 sen1 sen2 sen3 sen4 sen5
global A1 A2 A3 A4 A5
global a1 b1 c1 a2 b2 c2 a3 b3 c3 a4 b4 c4 a5 b5 c5
A=arduino('COM4');
A.pinMode(45,'output')
A.pinMode(47,'output')
A.pinMode(49,'output')
A.pinMode(51,'output')
A.pinMode(53,'output')
f = 0;
k = 0;

while f ~= 1
    k = k+1;
    if k >= v+1
        sen1(k) = A.analogRead(0);
        sen1 = sgolayfilt(sen1,n,v);
        sens(k,1) = round(sen1(k));
        ang1(k) = round((sens(k,1)^2)*a1 + sens(k,1)*b1 + c1);
        ang1 = round(sgolayfilt(ang1,n,v));

        sen2(k) = A.analogRead(1);
        sen2 = sgolayfilt(sen2,n,v);
        sens(k,2) = round(sen2(k));
        ang2(k) = round((sens(k,2)^2)*a2 + sens(k,2)*b2 + c2);
        ang2 = round(sgolayfilt(ang2,n,v));

        sen3(k) = A.analogRead(2);
        sen3 = sgolayfilt(sen3,n,v);
        sens(k,3) = round(sen3(k));
        ang3(k) = round((sens(k,3)^2)*a3 + sens(k,3)*b3 + c3);
        ang3 = round(sgolayfilt(ang3,n,v));

        sen4(k) = A.analogRead(3);

```

```
sen4 = sgolayfilt(sen4,n,v);
sens(k,4) = round(sen4(k));
ang4(k) = round((sens(k,4)^2)*a4 + sens(k,4)*b4 + c4);
ang4 = round(sgolayfilt(ang4,n,v));

sen5(k) = A.analogRead(4);
sen5 = sgolayfilt(sen5,n,v);
sens(k,5) = round(sen5(k));
ang5(k) = round((sens(k,5)^2)*a5 + sens(k,5)*b5 + c5);
ang5 = round(sgolayfilt(ang5,n,v));

if ang1(k)-105 > 69
    valslider1=69;
else
    if ang1(k)-105 < 0
        valslider1=0;
    else
        valslider1=ang1(k)-105;
    end
end

if ang2(k)-100 > 34
    valslider2=34;
else
    if ang2(k)-100 < 0
        valslider2=0;
    else
        valslider2=ang2(k)-100;
    end
end

if ang3(k)-98 > 82
    valslider3=82;
else
    if ang3(k)-98 < 0
        valslider3=0;
    else
        valslider3=ang3(k)-98;
    end
end

if ang4(k)-110 > 60
    valslider4=60;
else
    if ang4(k)-110 < 0
        valslider4=0;
    else
        valslider4=ang4(k)-110;
    end
end

if ang5(k)-110 > 60
    valslider5=60;
```

```

else
    if ang5(k)-110 < 0
        valslider5=0;
    else
        valslider5=ang5(k)-110;
    end
end

set(handles.slider1, 'Value', valslider1)
set(handles.slider2, 'Value', valslider2)
set(handles.slider3, 'Value', valslider3)
set(handles.slider4, 'Value', valslider4)
set(handles.slider5, 'Value', valslider5)

A1 = round(get(handles.slider1, 'Value')) + 105;
set(handles.text8, 'String', num2str(A1));
A2 = round(get(handles.slider2, 'Value')) + 100;
set(handles.text9, 'String', num2str(A2));
A3 = round(get(handles.slider3, 'Value')) + 98;
set(handles.text10, 'String', num2str(A3));
A4 = round(get(handles.slider4, 'Value')) + 110;
set(handles.text11, 'String', num2str(A4));
A5 = round(get(handles.slider5, 'Value')) + 110;
set(handles.text12, 'String', num2str(A5));
f=1;
else
    sen1(k) = A.analogRead(0);
    sens(k,1) = sen1(k);
    ang1 = (sens(k,1)^2)*a1 + sens(k,1)*b1 + c1;

    sen2(k) = A.analogRead(1);
    sens(k,2) = sen2(k);
    ang2 = (sens(k,2)^2)*a2 + sens(k,2)*b2 + c2;

    sen3(k) = A.analogRead(2);
    sens(k,3) = sen3(k);
    ang3 = (sens(k,3)^2)*a3 + sens(k,3)*b3 + c3;

    sen4(k) = A.analogRead(3);
    sens(k,4) = sen4(k);
    ang4 = (sens(k,4)^2)*a4 + sens(k,4)*b4 + c4;

    sen5(k) = A.analogRead(4);
    sens(k,5) = sen5(k);
    ang5 = (sens(k,5)^2)*a5 + sens(k,5)*b5 + c5;
end
end
end

```

```
% --- Executes during object creation, after setting all properties.
function text8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
end

% --- Executes during object creation, after setting all properties.
function text9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
end

% --- Executes during object creation, after setting all properties.
function text10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
end

% --- Executes during object creation, after setting all properties.
function text11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
end

% --- Executes during object creation, after setting all properties.
function text12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
end
```


C. Anexo: Manual de usuario para Interfaz de Control por Comandos de Voz

Con el fin de controlar el prototipo de prótesis de mano se construyó una interfaz gráfica de usuario en el entorno de programación Matlab®. A continuación se describe el funcionamiento y uso de dicha interfaz.

Apariencia de la Interfaz

La interfaz gráfica (Figura C. 1) posee cuatro botones de interacción, un área de visualización de ilustraciones, un área de visualización de instrucciones y un cuadro de texto que indica los comandos sugeridos por cada tipo de movimiento. El botón “Entrenar” inicia la secuencia de entrenamiento de la Red Neuronal mediante una rutina de grabación de comandos. El botón “Escuchar” activa la detección de comandos (definidos previamente en el botón Entrenar) a través de un micrófono para posterior accionamiento de la prótesis. El botón “Conectar” realiza la conexión entre el entorno Matlab® y la placa Arduino® a la cual se conectan los actuadores de la prótesis. Una vez se realiza exitosamente la conexión entre la interfaz gráfica y la tarjeta de control, el botón “Mover” se habilita y al presionar dicho botón, la prótesis ejecutará el movimiento correspondiente al comando detectado por el botón “Escuchar”.



Figura C. 1. Apariencia de la interfaz gráfica de control por comandos de voz.

Entrenamiento de la Red Neuronal Artificial

Al presionar el botón “Entrenar” (Figura C. 2) la interfaz indica en los cuadros de texto inferiores una serie de instrucciones que permiten al usuario grabar de manera exitosa los comandos necesarios para controlar la prótesis. El cuadro de texto más grande indica el comando que debe pronunciarse, dando la instrucción de que éste se vocalice luego de presionar la tecla *Enter*. El cuadro de texto más pequeño indica las repeticiones faltantes para determinado comando. El espacio de visualización central de la interfaz muestra una ilustración alusiva al comando que se está grabando en la RNA.



Figura C. 2. Funcionamiento del botón "Entrenar".



Figura C. 3. Resultado del entrenamiento de la RNA.

La interfaz mostrará un error calculado (Figura C. 3); si este error es inferior a 0,01, la RNA fue entrenada exitosamente. Si el error es mayor, se deberá repetir el proceso de entrenamiento para asegurar un buen funcionamiento del sistema.

Detección de comandos de voz

Una vez finalizado el entrenamiento, la interfaz estará lista para que el usuario vocalice comandos a ser detectados. Al presionar el botón escuchar, en el cuadro de texto inferior aparecerá el texto "Diga un comando" e inmediatamente el usuario deberá pronunciar el comando de seguridad ("Promanu"). Si la RNA detecta el comando "Promanu", en el cuadro de texto de menor tamaño aparecerá el comando de seguridad e inmediatamente se deberá vocalizar el comando correspondiente a la acción deseada.

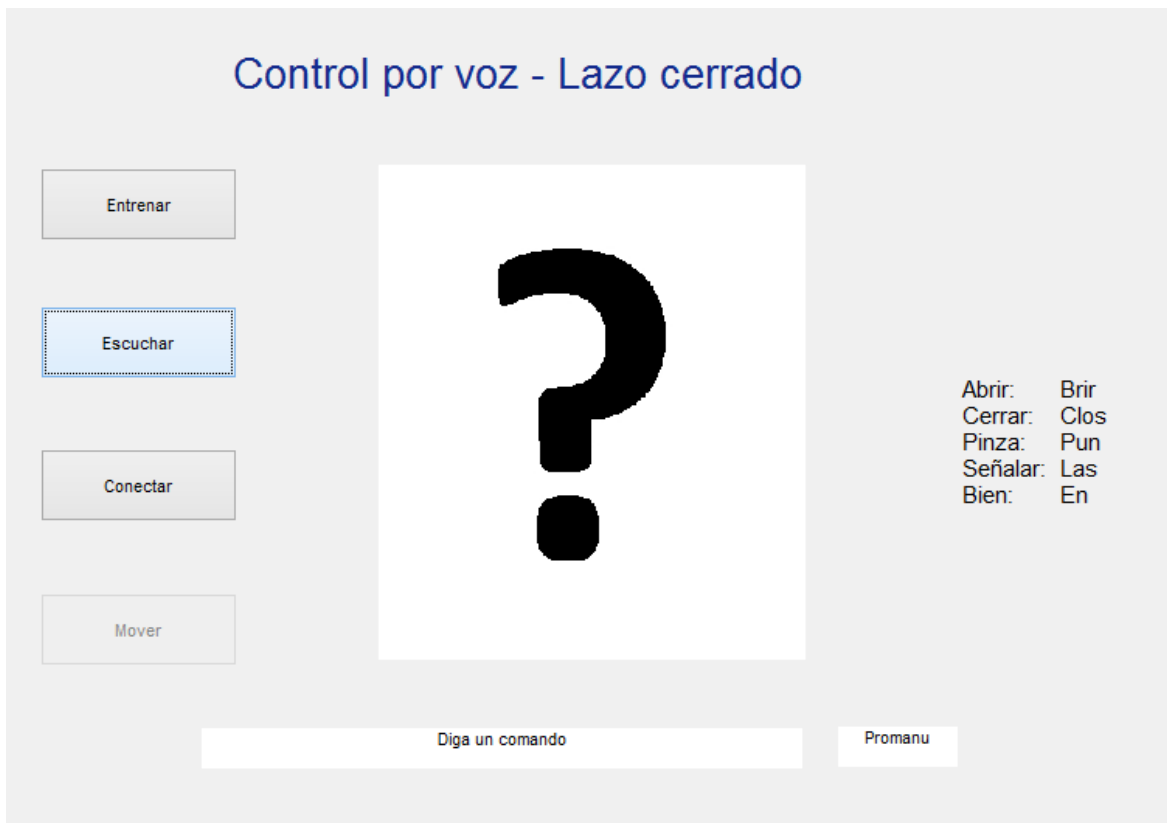


Figura C. 4. Funcionamiento del botón "Escuchar".

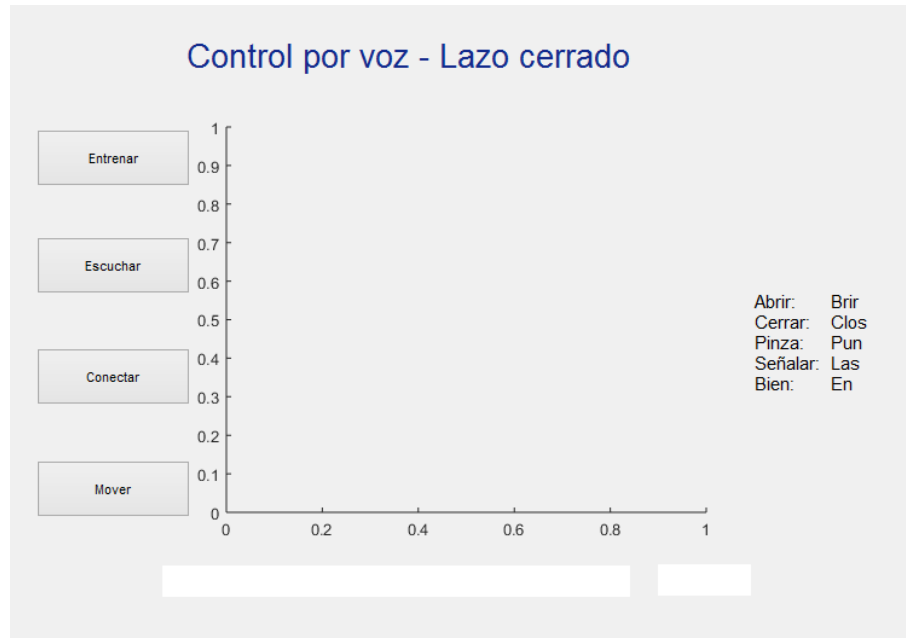
Si la RNA detecta o no el comando pronunciado por el usuario, la interfaz mostrará el comando identificado o un mensaje que indica que éste no fue detectado (Figura C. 5).



Figura C. 5. Ejemplo de detección de comandos. (a) No se ha detectado un comando. (b) Se detectó el comando "Cerrar".

Conexión y accionamiento de la prótesis

Para activar el enlace con el dispositivo es necesario seleccionar el botón “Conectar”. Si el enlace se realiza de forma exitosa, el botón “Mover” se activará, permitiendo enviar instrucciones al prototipo (Figura C. 6).



Cuando se ha detectado correctamente el comando de voz, el usuario podrá presionar el botón “Mover” de tal manera que se envíen las instrucciones requeridas al prototipo para lograr el movimiento o postura deseada (Figura C. 7).



Figura C. 7. Prototipo ejecutando la acción deseada.

Bibliografía

- [1] G. Ramón, «EL ESTUDIO Y ANÁLISIS DEL MOVIMIENTO HUMANO - Apuntes de Clase,» Instituto Universitario de Educación Física, Universidad de Antioquia (Colombia), Medellín, 2009.
- [2] A. Polanco y C. F. Rodriguez, «MODELO DINÁMICO DE MOVIMIENTOS HUMANOS,» de 8º CONGRESO IBEROAMERICANO DE INGENIERIA MECANICA, Cusco, 2007.
- [3] Naturalpoint, «Flex 3 - An affordable motion capture camera - OptiTrack,» Optitrack, [En línea]. Available: <http://www.optitrack.com/products/flex-3//>. [Último acceso: 11 Junio 2015].
- [4] L. Alonso Juan, «Análisis y control de un efector reproduciendo un movimiento circular de una mano,» Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET, Cuernavaca, México, 2006.
- [5] D. Pilaquina, «Diseño y Construcción de una Mano Robótica Controlada Mediante un Guante Sensorizado,» Escuela Politécnica Nacional, Quito, Ecuador, 2009.
- [6] J. Yang, E. Pena Pitarch, K. Abdel-Malek, A. Patrick y L. Lindkvist, «A multi-fingered hand prosthesis,» *Mechanism and Machine Theory*, nº 39, pp. 555-581, 2004.
- [7] I. Pérez y H. Mendoza, «DESARROLLO DE UNA MANO ROBÓTICA CON CAPACIDAD DE MANIPULACIÓN,» *Télématique: Revista Electrónica de Estudios Telemáticos*, vol. 7, nº 1, pp. 180-207, 2008.
- [8] R. Suárez y P. Grosch, «Mano Mecánica MA-I,» de XXIV Jornadas de Automática, León, España, 2003.
- [9] L. C. Sarmiento Vela, J. J. Páez y J. F. Sarmiento, «Prótesis mecatrónica para personas amputadas entre codo y muñeca,» *Tecné, Episteme y Didaxis*, nº 25, pp. 22-40, 2009.

- [10] M. H. Asyali, M. Yilmaz, M. Tokmakçi, K. Sedef, B. H. Aksebzec y R. Mittal, «Design and implementation of a voice-controlled prosthetic hand,» *Turkish Journal of Electrical Engineering and Computer*, vol. 19, nº 1, pp. 33-46, 2011.
- [11] Z. Feng, B. Yang, Y. Chen, Y. Zheng, T. Xu, Y. Li, T. Xu y D. Zhu, «Features extraction from hand images based on new detection operators,» *Pattern Recognition*, vol. 44, nº 5, pp. 1089-1105, 2011.
- [12] X. Yin y M. Xie, «Finger identification and hand posture recognition for human–robot interaction,» *Image and Vision Computing*, vol. 25, nº 8, pp. 1291-1300, 2007.
- [13] Y. Wu y T. Huang, «Hand Modeling, Analysis and Recognition,» *Signal Processing Magazine, IEEE*, vol. 18, nº 3, pp. 51-60, 2001.
- [14] E. Macho Stadler y M. J. Elejalde García, «La voz humana,» Universidad del País Vasco, 2002. [En línea]. Available: <http://ehu.eus/acustica/espanol/musica/vohues/vohues.html>.
- [15] J. Llisterri, «Métodos de análisis acústico del habla,» Universitat Autònoma de Barcelona, 10 Abril 2015. [En línea]. Available: http://liceu.uab.es/~joaquim/phonetics/fon_anal_acus/met_anal_acust.html.
- [16] M. H. S. Gülin Dede, G. Dede y M. Hüsnü Sazlı, «Speech recognition with artificial neural networks,» *Digital Signal Processing*, vol. 20, nº 3, 2010.
- [17] Practical Cryptography, «Mel Frequency Cepstral Coefficient (MFCC) tutorial,» [En línea]. Available: <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>. [Último acceso: Marzo 2015].
- [18] M.-L. Chen, S.-K. Changchien, X.-M. Zhang y H.-C. Yang, «The design of voice recognition controller via grey relational analysis,» *International Conference on System Science and Engineering (ICSSE)*, pp. 477 - 481, 2011.
- [19] J. Cheolwoo Jo, «Voice source simulator using Simulink and Matlab,» 2011.

-
- [20] F. F. Romero Fonseca y R. G. Álvarez Rueda, «Diseño y Construcción de un Sistema para el Control de Encendido y Apagado de Dispositivos Eléctricos por Medio de Comandos de Voz,» Escuela Politécnica Nacional, Quito, 2010.
- [21] H. Sakoe, S. Chiba y N. E. C. L. K. J. , «Dynamic Programming Algorithm Optimization for Spoken Word Recognition,» *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, nº 1, pp. 43-49, 2003.
- [22] R. Halavati, S. B. Shouraki y S. H. Zadeh, «Recognition of human speech phonemes using a novel fuzzy approach,» *Applied Soft Computing*, vol. 7, nº 3, pp. 828-839, 2007.
- [23] J. L. Loaiza Bernal, «Diseño y simulación de un prototipo de prótesis de mano bioinspirada con cinco grados de libertad,» Universidad Nacional de Colombia, Bogotá, 2012.
- [24] J. L. Loaiza Bernal y N. Arzola, «Modelamiento y diseño detallado de la prótesis de mano PROMANU,» de *XVI Convención Científica de Ingeniería y Arquitectura*, Ciudad de La Habana, 2012.
- [25] Naturalpoint, «Motion Capture Markers,» Optitrack, [En línea]. Available: <http://www.optitrack.com/products/motion-capture-markers/>. [Último acceso: 8 Julio 2015].
- [26] Naturalpoint, «Motive - Optical motion capture software,» Optitrack, [En línea]. Available: <http://www.optitrack.com/products/motive/>. [Último acceso: 8 Julio 2015].
- [27] J. Menger, «StickFigure - a C3D plotting and animation tool,» 5 Enero 2011. [En línea]. Available: <http://stickfigure.sourceforge.net/>.
- [28] B. Kollman y D. R. Hill, *Álgebra Lineal*, México: Pearson Education, 2006.
- [29] M. Weeks, *Digital Signal Processing using Matlab and Wavelets*, Hingham: Infinity Science Press LLC, 2007.
- [30] A. Ollero Baturone, *Robótica: manipuladores y robots móviles*, Barcelona: Marcombo, 2001.

- [31] S. Symbol, «Spectra Symbol Flex Sensor Datasheet,» [En línea]. Available: <http://www.spectrasymbol.com/wp-content/themes/spectra/images/datasheets/FlexSensor.pdf>. [Último acceso: 04 2014].
- [32] A. S. Sedra y K. C. Smith, *Microelectronic Circuits*, New York: Oxford University Press, 2004.
- [33] Arduino, «ArduinoBoardMega2560,» [En línea]. Available: <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>. [Último acceso: 25 11 2015].
- [34] Mathworks, «Arduino Support from MATLAB,» Mathworks, [En línea]. Available: <http://www.mathworks.com/hardware-support/arduino-matlab.html>. [Último acceso: 25 11 2015].
- [35] A. V. Oppenheim, A. S. Willsky y H. Nawab, *Señales y Sistemas*, México: Pearson Educación de México, 1998.
- [36] K. Ogata, *Ingeniería de Control Moderna*, Madrid: Pearson Educación, 2003.
- [37] U. A. Bakshi y A. P. Godse, *Analog And Digital Electronics*, Pune: Technical Publications, 2009.
- [38] B. Martín del Brío y A. Sanz Molina, *Redes Neuronales y Sistemas Borrosos*, Madrid: Ra-Ma, 2006.
- [39] Hilera y M. F. González, «Sistemas Conexionistas 1,» Universidad da Coruña, [En línea]. Available: <http://www.varpa.org/~mgpenedo/cursos/scx/archivospdf/Tema2-0.pdf>.
- [40] D. J. Matich, «Redes Neuronales: Conceptos Básicos y aplicaciones,» 2001.
- [41] M. A. Valencia Reyes, C. Yáñez y L. P. Sánchez, «Algoritmo Backpropagation para Redes Neuronales: conceptos y aplicaciones,» 2006. [En línea]. Available: <http://www.repositoriodigital.ipn.mx/bitstream/handle/123456789/8628/Archivo%20que%20incluye%20portada,%20%C3%ADndice%20y%20texto.pdf?sequence=1>. [Último acceso: Junio 2015].

- [42] V. Samant, «Voice Controlled Robot,» 11 01 2014. [En línea]. Available: <https://developer.mbed.org/users/vsamant3/notebook/mini-project-1--voice-controlled-robot/>.
- [43] R. Walpole, R. Myers, S. Myers y K. Ye, Probability and Statistics for Engineers and Scientists, Pearson Education, 2007.