UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Optimización del proyecto de una mano y muñeca animatrónica antropomórfica y su sistema de control y operación

Trabajo de graduación presentado por Miguel Alejandro García de León
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Optimización del proyecto de una mano y muñeca animatrónica antropomórfica y su sistema de control y operación

Trabajo de graduación presentado por Miguel Alejandro García de León
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2022

Vo.Bo.:

(f) 

Ing. Kurt Kellner

Tribunal Examinador:

(f) 

Ing. Kurt Kellner

(f) 

Ing. Luis Pinillos

(f) 

Ing. Pedro Castillo

Fecha de aprobación: Guatemala, 05 de enero de 2022.

Prefacio

Dedico este trabajo a Dios principalmente, a mis padres y hermanas por su apoyo durante todos estos años, por la formación personal que me brindaron y por ayudarme a cumplir mis metas.

Agradezco a la Fundación Juan Bautista Gutiérrez por la oportunidad que me brindó al otorgarme la beca para estudiar la carrera que me gusta y por el constante apoyo que me brindaron durante estos años.

Agradezco a mis compañeros por la constante motivación y apoyo durante todos estos años, por la ayuda que me brindaron y por todos los momentos que pasamos juntos.

Agradezco a mi asesor por su constante apoyo durante el proyecto y a los profesores del departamento de ingeniería mecánica por su apoyo.

Índice

Lista de figuras

Lista de cuadros

Resumen

El presente trabajo es una optimización del proyecto de una mano y muñeca animatrónica antropomórfica. En la fase anterior se realizó el diseño e implementación de una mano, muñeca y antebrazo animatrónico antropomorfo y se implementó un sistema de control sencillo por medio de MATLAB. El objetivo de este trabajo es optimizar el sistema de control, el movimiento de cada grado de libertad y la base del proyecto.

Se investigaron programas compatibles con el controlador del Leap Motion con el objetivo de implementar una interfaz gráfica de usuario. Se realizó un *trade study* con los distintos programas evaluando distintas características críticas para la selección y dándole un peso a cada una de ellas.

El programa ganador del estudio fue Unity. Se realizaron pruebas de detección de las manos, pruebas de comunicación serial, se crearon funciones para utilizar esta misma comunicación con botones de la interfaz. Utilizando programación orientada a objetos se encontró los ángulos de cada grado de libertad que posee la mano, se ajustaron, acotaron y se enviaron por medio de serial.

Con el objetivo de mejorar el movimiento de cada grado de libertad de la mano se analizó cada movimiento por separado. Se encontraron problemas con el largo de los tubos de teflón, problemas con la unión de cada hilo en la punta de los dedos, problema en la unión del tubo de teflón con la estructura de la palma y con el movimiento del dedo pulgar.

Se cambió parte del diseño de los dedos en el cual se agregó un pequeño agujero para colocar un pequeño clavo y así sujetar los hilos, se cambió el ángulo del compartimiento del hilo que extiende el dedo para mejorar el movimiento y se cambió el ángulo máximo de flexión en cada unión de las falanges.

Para resolver el problema de movimiento con el dedo pulgar se analizó el movimiento de los hilos desde cada posición que atraviesa. Se llegó a la conclusión de que el problema era la fricción de los dos hilos con el material de la estructura de la palma y para corregir esto se cambió de posición la entrada del tubo de teflón que mueve este dedo.

Utilizando como referencia los diseños de las yemas de los dedos elaborados por Omar Gálvez y el molde de las mismas, se rediseñó y fabricó el molde tomando en cuenta el paso de los hilos y su forma de unión. Se realizó una prueba con silicon y se propuso dos materiales para futuras pruebas.

Para que se pueda observar de una mejor forma todos los componentes del proyecto se rediseñó la base de la mano. Este nuevo diseño permite observar la electrónica, los servos y poleas que mueven cada grado de libertad y permite integrar el Leap Motion al proyecto utilizando una base y un disipador para el sensor.

Se rediseñó la muñeca para mejorar la apariencia de la mano utilizando engranes con relación 1 a 1 y colocando el servomotor del movimiento de desviación radial y cubital en la parte del antebrazo.

Abstract

The present work is an optimization of the project of an anthropomorphic animatronic hand and wrist. In the previous phase, the design and implementation of an anthropomorphic animatronic hand, wrist and forearm was carried out and a simple control system was implemented through MATLAB. The objective of this work is to optimize the control system, the movement of each degree of freedom and the structure of the project.

Programs compatible with the Leap Motion controller were investigated with the aim of implementing an user interface. A trade study was carried out with the different programs evaluating different critical characteristics for the selection and giving a weight to each one of them.

The winning program of the study was Unity. Hand detection tests were performed, serial communication tests, functions were created to use this same communication with buttons on the interface. Using object-oriented programming, the angles of each degree of freedom that the hand possesses were found, adjusted, dimensioned and sent by serial.

In order to improve the movement of each degree of freedom of the hand, each movement was analyzed separately. Problems were found with the length of the Teflon tubes, problems with the union of each thread at the fingertips, problems in the union of the Teflon tube with the palm structure and with the movement of the thumb.

Part of the finger design was changed in which a small hole was added to place a small metal nail to hold the threads, the angle of the thread compartment that extended the finger was changed to improve movement and the maximum angle of flexion at each joint of the phalanxes.

To solve the problem of movement with the thumb, the movement of the threads was analyzed from each position that it traversed. It was concluded that the problem was the friction of the two threads with the material of the palm structure and to correct this the inlet of the Teflon tube that moves this finger was repositioned.

Using as a reference the designs of the fingertips made by Omar Gálvez and their mold, the mold was redesigned and manufactured taking into account the pitch of the threads and their way of joining. A test was carried out with silicon and two materials were proposed for future tests.

In order to better observe all the components of the project, the base of the hand was redesigned. This new design allows to observe the electronics, the servos and pulleys that move each degree of freedom and allows to integrate the Leap Motion to the project using a base and a heatsink for the sensor.

The wrist was redesigned to improve the appearance of the hand using 1 to 1 ratio gearing and placing the servo motor of the radial and ulnar deviation movement in the part of the forearm.

CAPÍTULO 1

Introducción

Este proyecto tuvo como objetivo principal optimizar el diseño y control de la mano, muñeca y antebrazo animatrónicos construidos en el año 2020. Este documento presenta un marco teórico que ayuda a la comprensión del proyecto. Se presentan tres capítulos, los cuales son interfaz gráfica, diseño CAD y análisis de esfuerzos.

Para optimizar el control se implementó una interfaz gráfica de usuario en Unity con la finalidad de mejorar la experiencia entre el prototipo y el usuario.

Se rediseñaron las falanges para mejorar el movimiento y la unión de los hilos, se cambiaron las entradas de los tubos de teflón.

Para incorporar todos los elementos del prototipo se rediseñó una base que permite utilizarla en exposiciones y colocar e incorporar el Leap Motion en una posición óptima para poder utilizarlo y mostrar todos los componentes del proyecto.

Se rediseñó y fabricó el molde para fabricar las yemas de los dedos. Se realizaron pruebas con materiales flexibles, se fabricaron con colada de resina.

Se rediseñó la muñeca para mejorar la apariencia de la mano cambiando de posición el servo motor del movimiento de desviación radial y cubital.

Se realizó un análisis de elementos finitos en Autodesk Inventor.

CAPÍTULO 2

Antecedentes

El ser humano a lo largo del tiempo ha buscado entender como funciona el cuerpo humano y cómo está compuesto ya sea para poder encontrar soluciones a enfermedades o para poder replicarlo. El diseño de una mano robótica surge de necesidades cosméticas debido a la pérdida de la extremidad. Las primeras manos no presentaban ningún tipo de movimiento y eran para disimular la falta de la extremidad. Los avancen en el diseño de manos robóticas están ligados directamente en el desarrollo tecnológico, capacidad de manipulación de materiales y la comprensión de la anatomía humana [1].

En 2012 se desarrolló en la Universidad del Valle de Guatemala una prótesis biónica de una mano. Esta fase inicial se enfocó en el movimiento de los falanges y era transhumeral. Esta fase contaba con diversas desventajas, una de ella era que no tenía la capacidad de realizar movimientos individuales, el control era limitado y no tenía un diseño muy estético (Figura 1). A pesar de que esta fase se enfoca en una prótesis, sirve como precedente para las siguientes fases que se enfocan en una mano animatrónica [2].

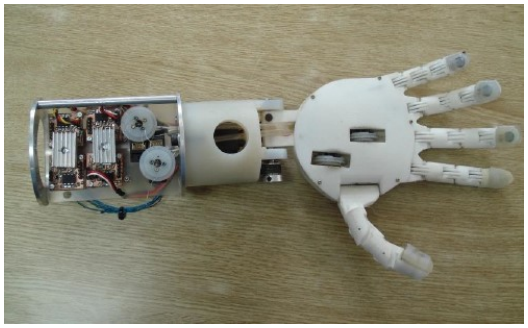


Figura 1: Mano elaborada por Pablo Mazariegos en 2012 [2].

Para la segunda fase, con base en la prótesis de la primera fase, se implementó una muñeca y el codo para un pirata animatrónico, proyecto que fue descontinuado y dividido en secciones para la siguiente fase. En la tercera fase se implementó un nuevo modelo de la

mano, mejorando los mecanismos de transmisión de las articulaciones y se trató de hacerla con un aspecto más humano (Figura 2). En esta fase se cambiaron los actuadores y se implementaron bisagras para las falanges [3].

Figura 2: Mano elaborada por Betti Rodas en 2019 [3].

Durante el año 2020 se optimizó el diseño de la mano animatrónica de la tercera fase utilizando fotografías de la mano autor con proyecciones en planos perpendiculares, se agregaron grados de libertad (antebrazo, muñeca y pulgar), se diseñó una base para la mano y se implementó el Leap Motion para el control de movimiento (Figura 3). En esta fase se mejoró el diseño de los tendones al utilizar tubos de teflón en configuración bowden [4].

Figura 3: Mano elaborada por Omar Gálvez en 2020 [4].

CAPÍTULO 3

Justificación

Los proyectos animatrónicos se han vuelto cada vez más populares ya que involucran diseños 3D, sistemas mecánicos, diseño electrónico y sistemas de control. Con el avance de la tecnología se encuentran mejores métodos para realizar esta clase de proyectos y tienen más realismo.

En la Universidad del Valle de Guatemala se han desarrollado proyectos animatrónicos desde los primeros cursos con proyectos sencillos hasta trabajos de graduación. La moyoría de estos trabajos de graduación se han quedado sin continuidad a lo largo de los años y algunos carecen de eficiencia en los diseños o de realismo.

Este trabajo se centra en optimizar el diseño anterior de la mano animatrónica con el objetivo de mejorar el movimiento de los dedos y mejorar la apariencia de la mano, así mismo integrar un mejor sistema de control del que se posee ya que este es limitado y funciona durante solo un período de tiempo.

Este trabajo es de gran importancia ya que podré implementar todos los conocimientos adquiridos a lo largo de la carrera universitaria integrando desde diseño y fabricación 3D, programación e implementación de sistemas de control.

Este trabajo de graduación se podrá exponer en diversas actividades por parte del departamento de Ingeniería Mecatrónica, Electrónica y Biomédica para motivar a los futuros estudiantes de las carreras antes mencionadas.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Optimizar el diseño y control de la mano, muñeca y antebrazo animatrónicos construidos en la Universidad del Valle de Guatemala.

4.2. Objetivos específicos

- Diseñar una interfaz gráfica de usuario intuitiva para uso en demostraciones públicas.
- Optimizar el sistema de control y diseño mecánico para lograr un movimiento fluido en todos los dedos de la mano y la muñeca.
- Diseñar una base rígida y estable para el proyecto que incluya los sensores de interacción con el usuario.
- Fabricar e instalar la yema de los dedos con un material flexible.
- Realizar análisis de elementos finitos a los nuevos diseños propuestos.

CAPÍTULO 5

Alcance

Este trabajo contiene optimizaciones de la mano y muñeca animatrónica antropomórfica y su sistema de control y operación.

Con las pruebas realizadas en cada grado de libertad de la mano y observando detalladamente cada ángulo de la trayectoria del tubo de teflón se logró realizar el movimiento en el dedo pulgar y se mejoró el movimiento en el resto de los dedos.

Gracias a la investigación realizada y al trade study hecho, se desarrolló una interfaz gráfica de usuario que permite al usuario interactuar con el prototipo.

Se rediseñó la base para el proyecto y se fabricó utilizando los materiales propuestos. Se utilizó MDF para la estructura y acrílico para poder observar los componentes internos de la mano. Se implementó un disipador de calor y un soporte para el LEAP Motion con distintos ángulos de soporte.

Se fabricaron las yemas de los dedos, se realizaron dos pruebas para definir el material a utilizar, siendo el material elegido Platinum Silicone Rubber utilizando el método de colada de resina para fabricar moldes.

Se rediseñó la muñeca mejorando el aspecto del prototipo cambiando de posición un servomotor y utilizando engranajes con relación 1 a 1 y con el mayor número posible de dientes para logran un movimiento preciso.

Se realizó un análisis de elementos finitos en Autodesk Inventor para validar el diseño.

CAPÍTULO 6

Marco teórico

La mano es de los mecanismos más complejos debido a la gran cantidad de movimientos posibles además de la precisión de estos, al ser la responsable de la completa manipulación de objetos es uno de los proyectos más populares en la animatrónica [5].

6.1. Morfología y fisiología de la mano

6.1.1. Tendón

Órgano formado por haces de tejido fibroso, de color blanco brillante y muy resistentes a la tracción, que por lo común unen los músculos a los huesos [6].

6.1.2. Metacarpo y carpo

La mano consta de distintos huesos, el metacarpo son los huesos que unen las falanges con los huesos del carpo, el carpo son los huesos que unen el metacarpo con la muñeca (huesos: radio y cubito) [7].

Los huesos del metacarpo conforman la mayor parte de la palma, estos son cinco huesos largos y se relacionan con cada uno de los dedos: I Metacarpiano (pulgar), II Metacarpiano (índice), III Metacarpiano (medio), IV Metacarpiano (anular) y V Metacarpiano (meñique) [7].

Figura 4: Huesos de la mano, vista palmar [8].

Los huesos del carpo son 8 y se clasifican en dos grupos: huesos de la fila proximal o primera fila del Carpo: Escafoides, Semilunar, Piramidal y Pisiforme; y huesos de la fila distal o segunda fila del carpo; Trapecio, Trapezoide, Grande, Ganchoso. Estos huesos se pueden observar en la Figura 5 [7].

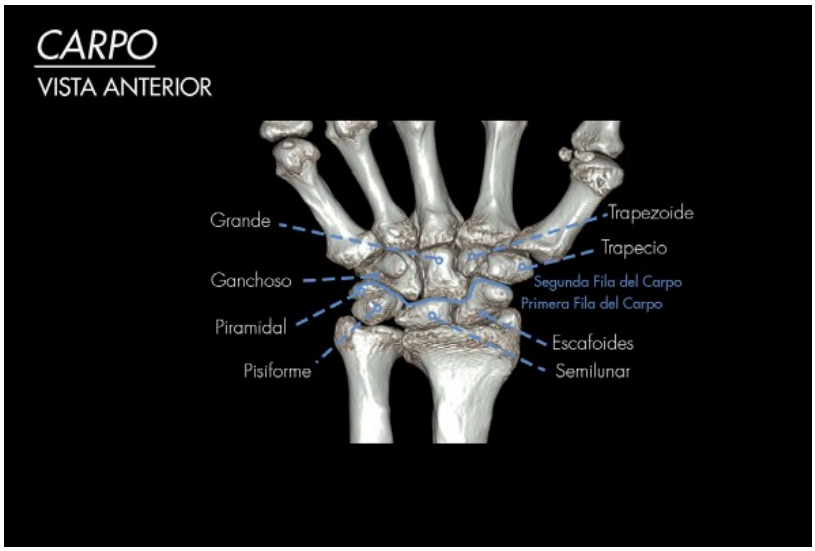


Figura 5: Huesos del carpo, vista anterior [8].

6.1.3. Falanges

Los dedos índice, medio, anular y meñique poseen movimiento de flexión y extensión (Figura 6) y el dedo pulgar posee un movimiento más complejo que incorpora movimientos en el metacarpo [7].

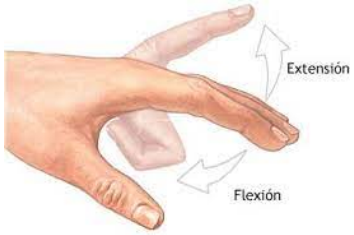


Figura 6: Movimiento de los dedos [9].

Las falanges son los huesos de los dedos, el pulgar posee dos falanges, proximal y distal, y el resto de dedos posee tres falanges, proximales, medios y distales, estos huesos se pueden ver en la Figura 4. El falange proximal es el más largo y más grande y el falange distal es el más corto y más pequeño [7].

La falange proximal se articula con la cabeza del metacarpiano asociado en cada articulación metacarpofalángica y con base en la falange media asociada en la articulación interfalángica proximal [7].

La articulación metacarpofalángica es una articulación multiaxial que permite el movimiento en las direcciones medial y lateral, y una ligera rotación. La articulación se estabiliza por los ligamentos colaterales, placa volar, articulación cápsula y tendones superpuestos intrínsecos y extrínsicos (Figura 7) [7].

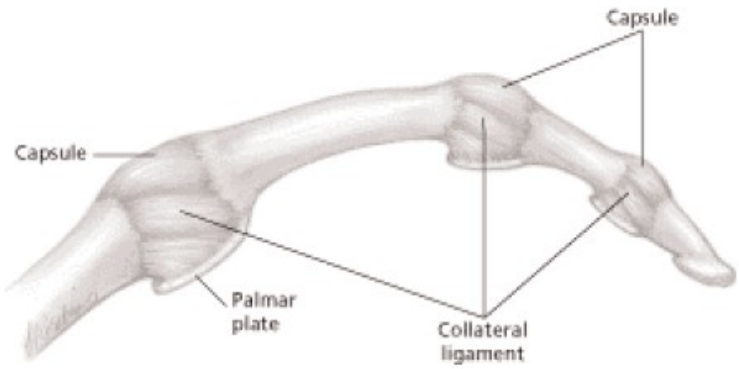


Figura 7: Vista lateral de las articulaciones interfalángicas [8].

La base de la falange distal se articula con la cabeza de la falange media por medio de una articulación intefalángica, esta es una articulación en bisagra. La articulación se estabiliza por los ligamentos colaterales, placa volar y los tendones extrínsecos del flexor profundo [7].

6.1.4. Pulgar

La base de la falange proximal es similar a la falange proximal de los otros. La base de esta falange posee una ligera cresta o área rugosa que separa la superficie articular del eje, esta área rugosa proporciona el lugar de intersección del extensor corto del pulgar (Figura

8), el extensor largo se inserta en la base del distal. En la superficie palmar de la base de la falange proximal hay un pequeño surco para acomodar el tendón flexor. La superficie articular en la base es más plana y menos cóncava para acomodar la superficie articular de la cabeza del metacarpiano del pulgar, que tiende a ser más plana y menos esférica que en los otros metacarpianos [7].

Figura 8: Extensor corto del pulgar [10].

La articulación metacarpofalángica (flexión de 75° a 80° y extensión 0°) es similar a la de las otras articulaciones metacarpofalángicas. Sin embargo, debido a la forma de las superficies articulares contiguas, la articulación es más similar a una bisagra en lugar de multiaxial como en las otras. La articulación está estabilizada por ligamentos colaterales, junto con los músculos intrínsecos y los tendones extrínsecos suprayacentes (flexor largo del pulgar y extensor largo del pulgar) [7].

La articulación interfalángica (flexión de 75° a 80° y extensión de 5° a 10°, puede llegar a 30°) es una articulación en bisagra, más grande que las articulaciones interfalángicas de los otros. Está estabilizado por los ligamentos colaterales, la placa volar y los tendones extrínsecos suprayacentes [7].

Figura 9: Extensor largo del pulgar [10].

Varios músculos se insertan en la base de la falange proximal del pulgar. El abductor corto (Adductor pollicis) se inserta en la cara radial de la base (Figura 10). El flexor corto se inserta en la base palmar del proximal y el flexor largo se inserta en la base palmar del distal. El aductor del pulgar se inserta en la cara cubital de la base [7].

El pulgar tiene ocho músculos motores, distribuidos en dos grupos:

Músculos extrínsecos:

- Abductor largo: lleva el primer metacarpiano hacia fuera y hacia delante, por lo que es abductor y flexor (Figura 10) [11].

- Extensor corto: extiende la articulación metacarpofalángica y lleva el pulgar hacia fuera, es por lo tanto es el verdadero abductor del pulgar (Figura 8) [11].

- Extensor largo: extiende la articulación interfalángica y pasivamente lleva el metacarpiano y la falange proximal hacia adentro y atrás, contribuyendo a aplanar la palma de la mano (Figura 9) [11].

- Flexor largo: flexiona la articulación interfalángica y pasivamente produce flexión de la falange proximal (Figura 10) [11].

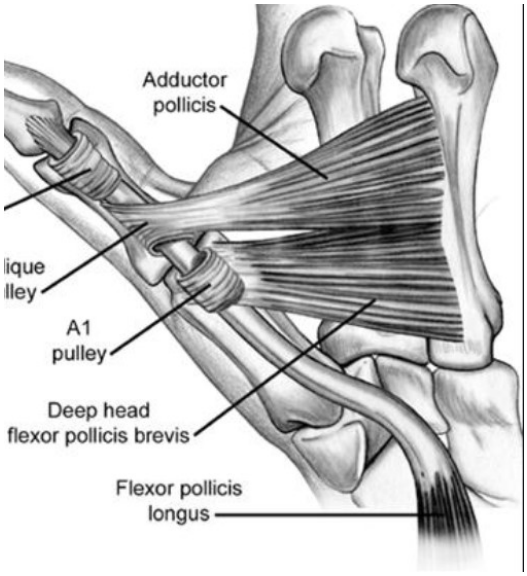


Figura 10: Abductor corto y largo, y Flexor largo. Vista palmar [10].

Músculos extrínsecos:

- Flexor corto: flexiona la falange proximal sobre el primer metacarpiano (Figura 11) [11].
- Oponente: flexiona el primer metacarpiano sobre el carpo, aducción del primer metacarpiano y rota axial-mente [11].
- Abductor corto: produce flexión con inclinación radial de la falange proximal sobre el primer metacarpiano [11].
- Aductor: genera flexión ligera con inclinación cubital de la falange proximal sobre el primer metacarpiano [11].

Figura 11: Flexor corto, vista palmar [10].

6.2. Morfología y fisiología de la muñeca y antebrazo

El antebrazo está formado pos dos huesos principalmente: el radio y el cubito. Estos huesos se pueden observar en la Figura 12.

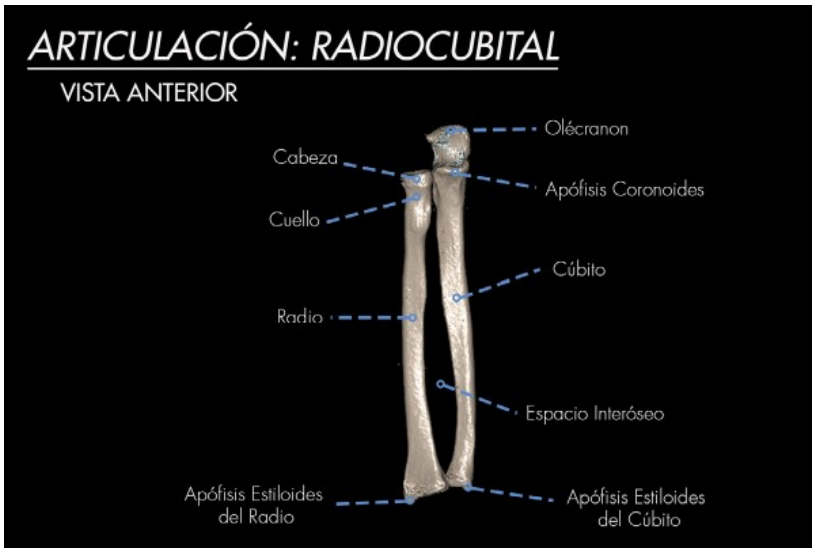


Figura 12: Huesos del antebrazo, vista anterior [8].

6.2.1. Antebrazo (articulación radiocubital)

El antebrazo consta de dos movimientos: supinación y pronación (Figura 13), es un movimiento conoide de base distal (el radio gira sobre el cúbito y este sobre su eje). En la articulación radiocubital proximal durante la supinación los huesos están paralelos mientras en la pronación el radio se coloca por encima del cubito [12].

Figura 13: Movimientos del antebrazo [12].

6.2.2. Muñeca

Consta de cuatro movimientos (Figura 14):

- Flexión: es de 90° máximo, aunque solo se utilice un rango de movilidad de 10° a 15° parar actividades de la vida diaria, la palma se inclina en dirección anterior del antebrazo, se reducirá el movimiento si los dedos están en posición de flexión [13].
- Extensión: es de 80° máximo, a pesar de que en las actividades de la vida diaria sólo se utiliza una amplitud de movimiento de 35°, el dorso de la mano se inclina en dirección posterior del antebrazo, se reducirá el movimiento si los dedos están en posición de extensión [13].
- Desviación radial o Abducción: es de 20° grados máximo y la mano se inclina hacia el radio [13].
- Desviación cubital o Aducción: es de 35° grados máximo y la mano se inclina hacia el cubito [13].

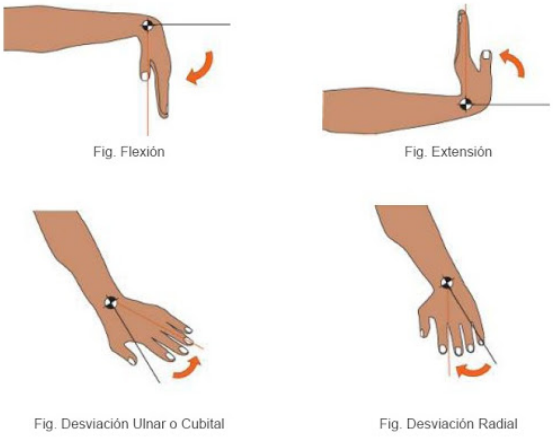


Figura 14: Movimientos de la muñeca [13].

6.3. Diseño CAD

- CAD (Computer Aided Design): Diseño asistido por computador [14].
- CAM (Computer Aided Manufactoring): Fabricación asistida por computador [14].
- CAE (Computer Aided Engineering): Ingeniería asistida por computador [14].

Actualmente CAD se le comprende como la integración del CAD y CAE [14].

6.3.1. Fusion 360

Es un software CAD, CAM y CAE basado en almacenamiento en la nube, combina el diseño industrial y mecánico, simulación, colaboración y maquinado, todo en uno. Las herramientas en Fusion 360 permiten la exploración rápida y fácil de ideas para hacerlas realidad. Este programa permite importar distintos tipos de archivos como OBJ y STL [15].

6.4. Leap Motion Controller

Es un módulo óptico de seguimiento manual que captura los movimientos de las manos con una gran precisión. Permite la interacción entre el usuario y el ordenador mediante gestos en el espacio. Este dispositivo consta de dos cámaras monocromáticas de 640x240 píxeles a 120Hz con tres LEDs infrarrojos [16].

Este módulo óptico de seguimiento manual de alta precisión ofrece un campo de visión de 135° y un rango de hasta 80cm, puede rastrear objetos y capturar imágenes infrarrojas de alta velocidad y este dispositivo puede interactuar con contenido digital y con aplicaciones VR y AR [16].

Figura 15: LEAP Motion Controller [16].

El software de Leap Motion es capaz de distinguir 27 elementos distintos de la mano, incluidos huesos y articulaciones, y rastrearlos incluso cuando están ocultos por otras partes de la mano. Las dimensiones de este dispositivo son bastante reducidas en comparación con otros interfaces gestuales del mercado: tan solo mide alrededor de 80 mm de largo, 30 mm de profundidad y 11 mm de alto [16].

Medición	**Unidades**
Distancia	Milímetros
Tiempo	Microsegundos
Velocidad	Milímetros/segundos
Ángulo	Radianes

Cuadro 1: Unidades del LEAP Motion [16].

Cuando un objeto, como una mano en este caso, es iluminado, se produce una reflexión de luz que llega al dispositivo e incide sobre las lentes de las dos cámaras. Estas lentes, de tipo biconvexas, concentran los rayos en el sensor de cada cámara; y los datos recogidos por los sensores se almacenan en una matriz en la memoria del controlador, en donde se realizan los ajustes de resolución adecuados mediante el microcontrolador del dispositivo [17].

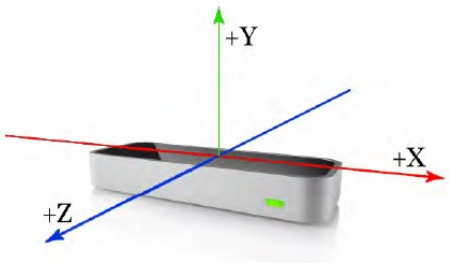


Figura 16: Sistema de coordenadas del LEAP Motion [16].

El sistema de coordenadas se muestra en la Figura 16. Además, la API del LEAP Motion utiliza las unidades del Cuadro 1. Los datos que se extraen del controlador, tanto de velocidad como de posición u orientación, en la Figura 17 se observan todos los puntos que se monitorizan [17].

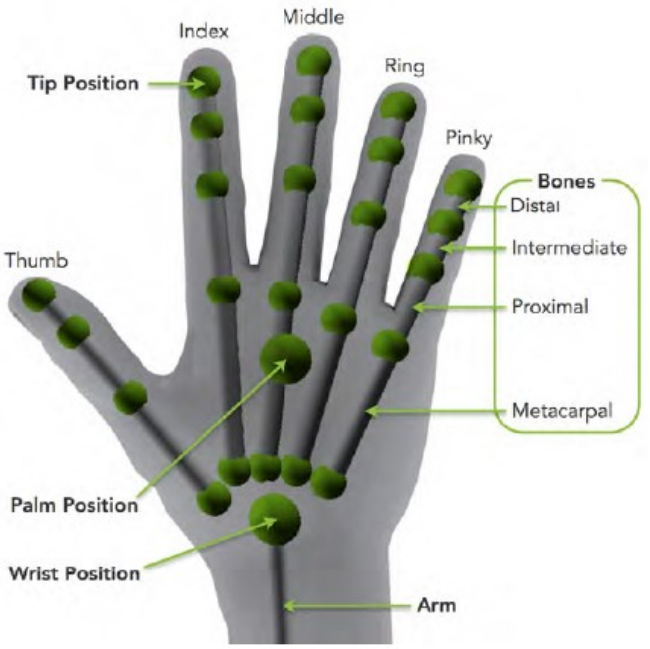


Figura 17: Estructura de la mano usada en el LEAP Motion [16].

6.5. Interfaz gráfica de usuario

6.5.1. Unity

Es un software de desarrollo de videojuegos, experiencias de realidad virtual, miniseries o interfaces gráficas en tiempo real creado por Unity Technologies. Una de las características más importantes de Unity es que soporta la exportación a una enorme cantidad de plataformas. Esta herramienta engloba motores para renderizar imágenes, motores de audio y motores de animación [18].

Esta herramienta además de crear videojuegos, que es el uso más extendido, es capaz de crear catálogos de productos en realidad aumentada, entornos virtuales interactivos para arquitectura, aplicaciones para móviles o eventos con experiencias de realidad virtual, entre otras cosas [18].

Figura 18: Unity Logo [18].

6.5.2. Unreal

Es un entorno de desarrollo creado por la compañia Epic Games que incluye todas las herramientas necesarias para construir una simulación: editor de vídeo, estudio de sonido, renderización de animaciones, etcétera. Con su código escrito en C++, el Unreal Engine presenta un alto grado de portabilidad y es una herramienta utilizada actualmente por muchos desarrolladores de juegos [19].

Originariamente, se creó como motor de juegos para programadores, pero gracias a su versatilidad, poco a poco se ha ido haciendo un espacio en sectores tan diversos como la arquitectura, ingeniería, medicina o realidad virtual. Con Unreal Engine se pueden crear paisajes, entornos interactivos o realidad virtual; desde videojuegos en 2D hasta experiencias completamente inmersivas [19].

Figura 19: Unreal Engine Logo [19].

6.5.3. LeapC API

Es una interfaz de programación de aplicaciones en lenguaje C para acceder a los datos de movimiento desde el Leap Motion. La biblioteca LeapC actúa como intermediario entre la aplicación y el servicio Leap Motion. LeapC implementa una arreglo de mensajes a la que publica mensajes de seguimiento, imagen y estado desde el servicio [20].

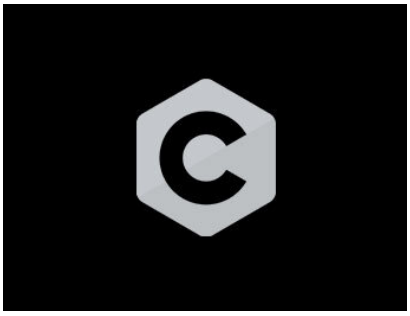


Figura 20: LeapC API [20].

6.5.4. QT

Es un entorno de trabajo multiplataforma orientado a objetos utilizado mayormente para desarrollar software que utilicen interfaz gráfica de usuario. Utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios otros lenguajes de programación. También se usa en sistemas informáticos embebidos para automoción, aeronavegación y aparatos domésticos [21].

Figura 21: QT logo [21].

CAPÍTULO 7

Diseño mecánico

7.1. Falanges

Se encontró problemas en la unión del hilo que simulan ser los tendones flexores y extensores, estos al estar solo pegados no garantizaban tener un buen agarre. Algunos se despegaron y causaron un tipo de juego en el movimiento, eso se puede observar en la Figura 22.

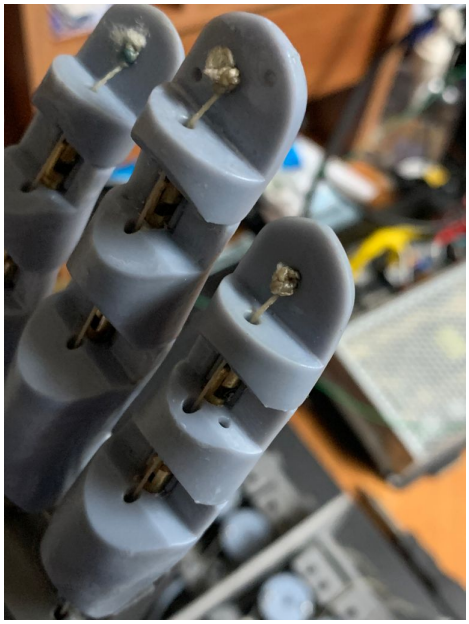


Figura 22: Unión entre hilos.
Elaboración propia.

Otro problema que se encontró fue que el dedo al bajar completamente provoca un ángulo de 90° entre la falange proximal y la palma (Figura 23) por lo que causa que el servo haga una gran fuerza para levantarse y el movimiento no sea fluido.

Figura 23: Error de extensión.
Elaboración propia.

Para solucionar este problema se rediseñaron los dedos y fabricaron reduciendo los ángulos entre las uniones de las falanges como se muestra en la Figura 24.

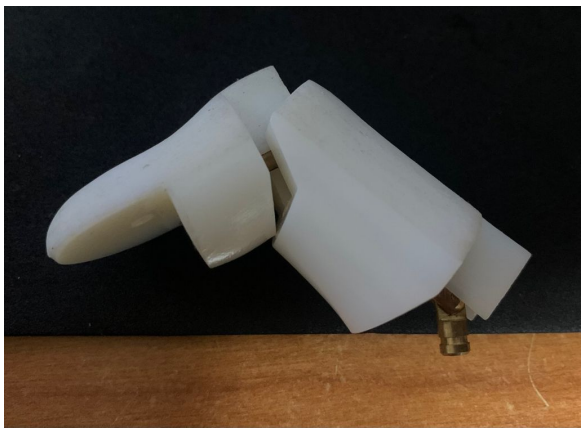


Figura 24: Propuesta de falanges.
Elaboración propia.

Con el objetivo de mejorar un poco más el movimiento de los dedos se cambió el ángulo del conducto del hilo que extiende los dedos, esto para no tener ángulos de 90° al estar completamente flexionados los dedos (Figura 26).

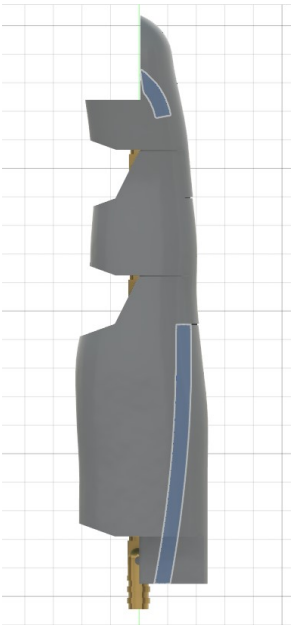


Figura 25: Propuesta de conducto de extensión.
Elaboración propia.

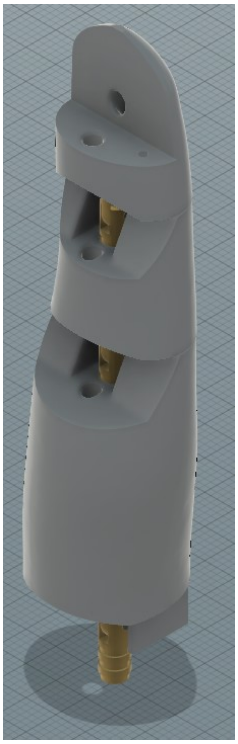


Figura 26: Propuesta de dedo completa.
Elaboración propia.

Al momento de flexionar los dedos estos caían de una forma brusca por su propio peso y al momento de levantarse el servo motor apenas lograba levantar por completo el dedo. Para

solucionar este problema se agregó un pequeño resorte con alta resistencia a la fatiga para lograr un movimiento más controlado de flexión y que al mismo tiempo ayude a extender el dedo. Esto se puede observar en la Figura 27.

Figura 27: Resorte en falange.
Elaboración propia.

7.2. Dedo pulgar

Al igual que en los otros dedos se encontró un problema en la unión del hilo en la yema de los dedos. Para solucionar el problema se propuso agregar un elemento para sujetar y amarrar los hilos, la primera propuesta fue un tornillo pero este al tener que ser muy pequeño por las dimensiones de la falange distal el precio por unidad es muy elevado.

La segunda propuesta fue utilizar un clavo y doblarlo para no ocupar demasiado espacio como se observa en la Figura 28. Esta propuesta se fabricó y se probó.

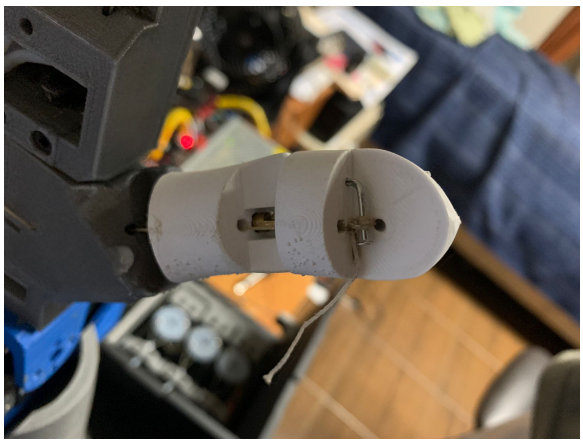


Figura 28: Prueba de amarre de hilos en falange distal.
Elaboración propia.

Se probó el movimiento del dedo solo con los hilos desde la parte final del recorrido de este, se tuvo que hacer una fuerza grande y solo se movió la falange distal. Se hicieron pruebas moviendo los hilos desde la cavidad del metacarpo y este se movió con facilidad, luego desde la palma y el dedo se movió con un poco mas de dificultad. El problema que se encontró fue que los hilos tienen un recorrido sin el tubo de teflón (en este se encuentran lubricados) y estos al estar tensionados provocan una gran fuerza de fricción por lo que sería muy complicado mover el dedo.

Para demostrar esto se cambió la posición de entrada del tubo de teflón como se observa en la Figura 29 y se probó el movimiento siendo este muy fluido y sin utilizar gran fuerza.

Figura 29: Prueba 1 de movimiento de pulgar.
Elaboración propia.

Con el resultado anterior se rediseñó el compartimiento del metacarpo permitiendo ingresar el tubo teflón desde el mismo y se agregó un agujero para sujetar los hilos que mueven al metacarpo ya que al igual que todos los demás hilos solo estaban pegados.

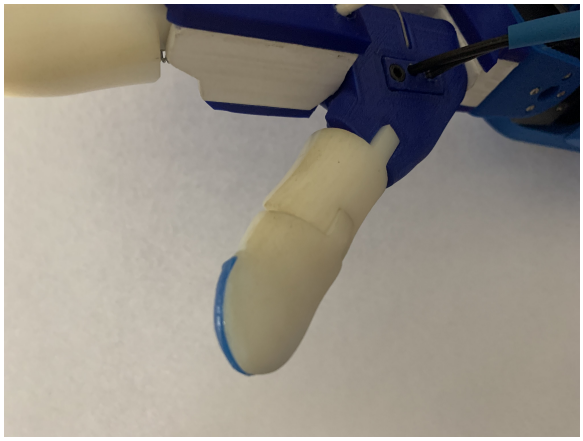


Figura 30: Propuesta de dedo completa.
Elaboración propia.

7.3. Estructura de la palma de la mano

Al analizar la estructura de la palma (Figura 31) se encontraron algunos problemas con los tubos de teflón. El tubo del dedo índice se encontraba en una mala trayectoria y los tubos no se unían de una forma correcta a la estructura.

Figura 31: Estructura de la palma.
Elaboración propia.

Se rediseñó y fabricó una propuesta (Figura 32) en la que el tubo del dedo índice entra bajo el mismo, se agregó un componente para la unión de los tubos y la estructura, y se agregó chaflanes en algunas esquinas para mejorar la trayectoria del tubo. Además, se cambió

el ángulo superior de la palma para que la falange proximal no tenga un ángulo de 90° con la palma.

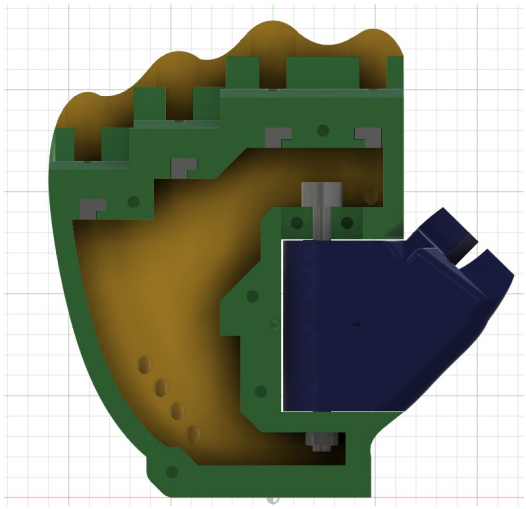


Figura 32: Propuesta de estructura de la palma.
Elaboración propia.

7.4. Base para proyecto

Se rediseñó la base para el proyecto con el objetivo de que se pueda observar los componentes electrónicos y el sistema de poleas con los servos. Para ello la parte trasera de la base se diseñó con material acrílico y el resto con MDF. Se dejó un espacio del lado izquierdo de la base para incorporar el Leap Motion.

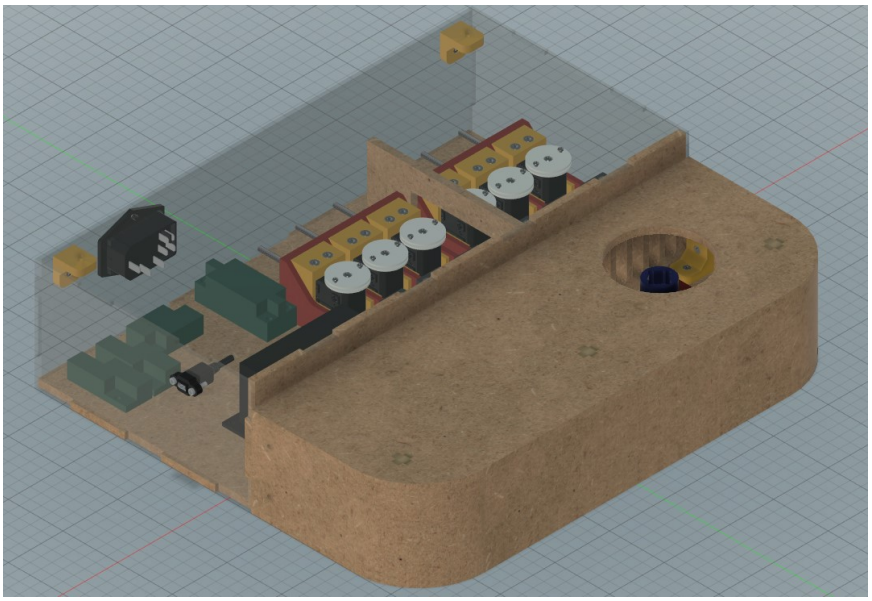


Figura 33: Propuesta de estructura de la palma.
Elaboración propia.

Al utilizar el Leap Motion durante un periodo largo de tiempo se notó que este se calentaba bastante y que su rendimiento disminuía levemente. Para corregir un poco el efecto de la temperatura se porpuso agregarle una base de aluminio para utilizarlo como disipador de calor.

Se utilizó un sensor infrarrojo de temperatura MD-GY-906 y un arduino uno para medir la temperatura del Leap Motion después de utilizarlo unos 20 minutos y este mostró temperaturas de 45°C aproximadamente. Luego se colocó el Leap Motion sobre un bloque de aluminio y se realizó la misma prueba dando como resultado temperaturas entre 30 y 34°C. Todos estos análisis se hicieron a una temperatura ambiente de 24 °C.

Se propuso un diseño para el bloque de aluminio que disipará el calor del sensor:

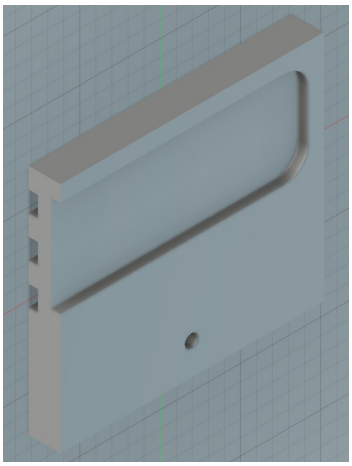


Figura 34: Disipador para el Leap Motion.
Elaboración propia.

Para unir el disipador y la base del proyecto se propuso una base (Figura 36) que nos permita colocar este en distintas posiciones ya que al momento de exponer el proyecto en varios lugares no se puede asegurar la misma altura y posición. Se propuso una base con ángulos de 35, 50, 60 y 45. Esta base se une con el disipador por medio de un tornillo.

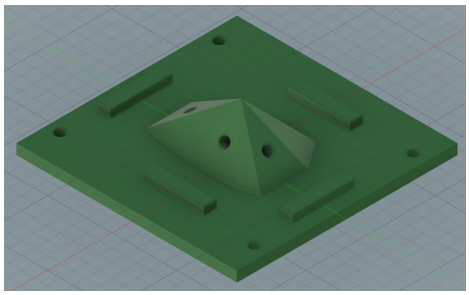


Figura 35: Base para el disipador.
Elaboración propia.

Luego de analizar la forma en que se une la base del disipador a la base del proyecto, esta al utilizarla se pierde mucho tiempo en poner y quitar los tornillos y cambiar de posición

la dirección de las cámaras del LEAP Motion de adelante hacia atrás. Se modificó la base del disipador eliminando los tornillos y agregando una base con la que solo se introduzca deslizando la pieza.

Con estas modificaciones se reduce el tiempo para cambiar de dirección horizontal las cámaras del sensor y dejando únicamente un tornillo para cambiar el ángulo vertical de las cámaras. Se fabricaron las piezas (disipador y bases) y se montaron, se probó su funcionamiento y dio un resultado exitoso.

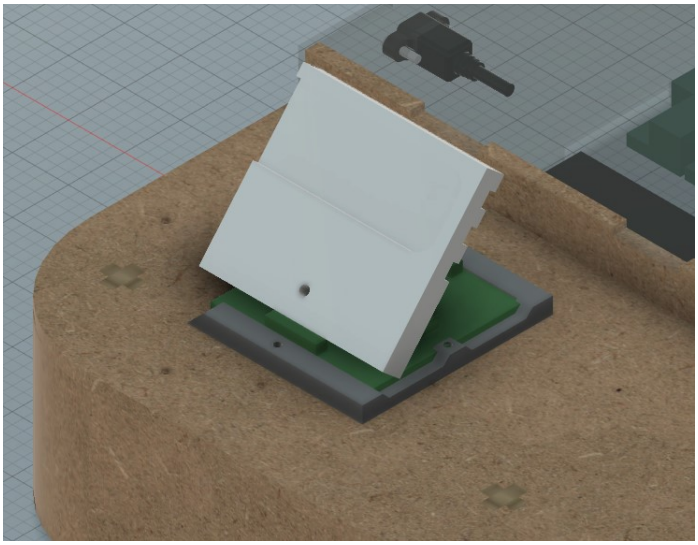


Figura 36: Segundo prototipo para la base para el disipador.
Elaboración propia.

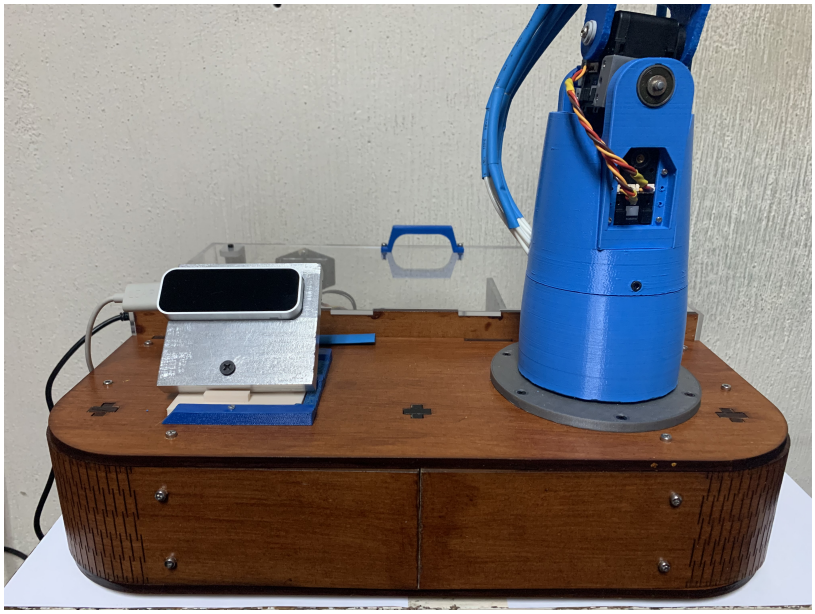


Figura 37: Base terminada y armada.
Elaboración propia.

Figura 38: Parte trasera de la base.
Elaboración propia.

7.5. Muñeca

Al analizar el movimiento de la muñeca se encontró que los tubos de teflón son muy cortos y reducen la movilidad de la mano.

Figura 39: Tubos de teflón muy cortos.
Elaboración propia.

Para mejorar la apariencia de la muñeca, esta se rediseñó cambiando de posición el servo que genera el movimiento de desviación radial y cubital ubicándolo debajo del servo que genera el movimiento de flexión y extensión. Se utilizó un sistema de engranajes diseñándolos con base en la distancia entre ejes de los servos. Para lograr un movimiento más preciso se utilizó 28 dientes para cada engrane (relación 1 a 1) y se utilizó 10mm de ancho de cara para lograr estabilidad y reducir esfuerzos de contacto.

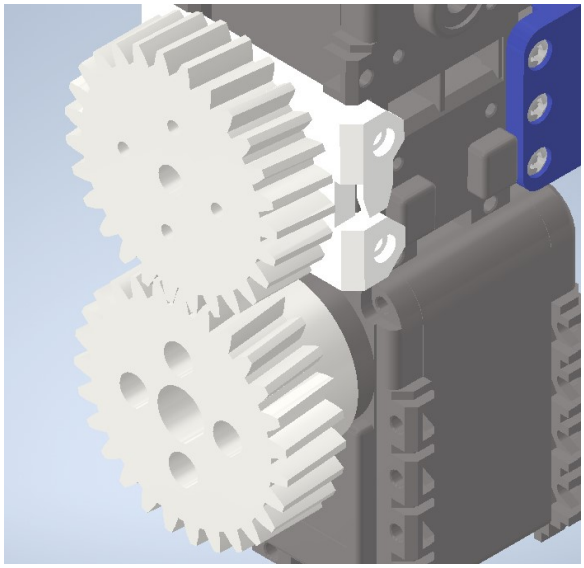


Figura 40: Engranes de la muñeca.
Elaboración propia.

Los engranes se diseñaron con cuatro agujeros para poder acoplarlos a los servos. Se reutilizó el sistema de eje con tornillo y rodamiento para el eje de la muñeca y se replicó del otro lado para completar el eje.

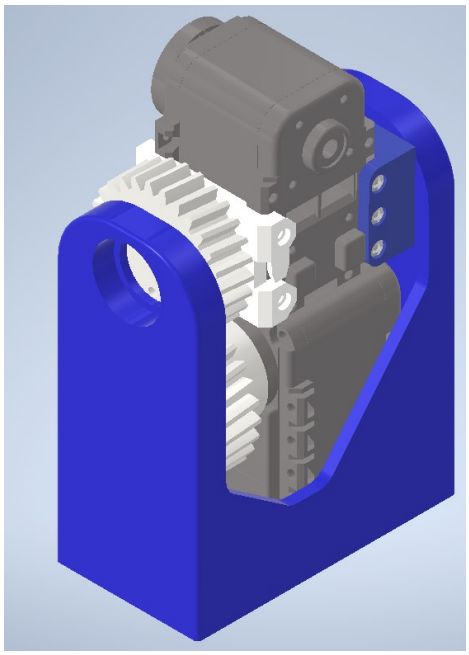


Figura 41: Nuevo diseño de la muñeca.
Elaboración propia.

Se fabricó el prototipo, se montó y probó dando un resultado exitoso. Con este prototipo se mejora la apariencia de la mano logrando que esta tenga un mayor parecido a una mano

real.

Al ser esta muñeca más alta y con menos profundidad se rediseñó el antebrazo para reducir la altura del antebrazo. El resultado puede observarse en la Figura 42.

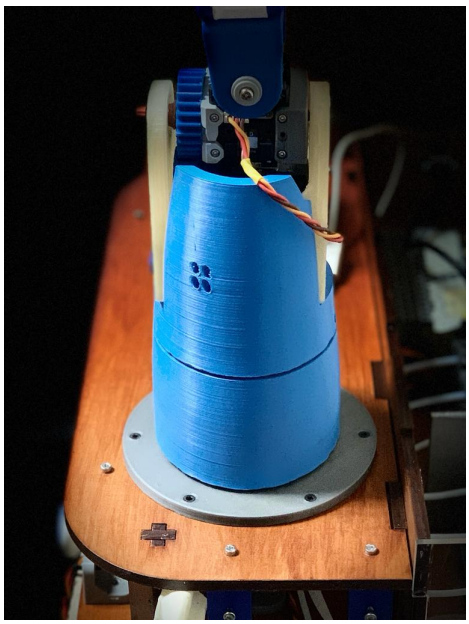


Figura 42: Antebrazo y muñeca rediseñadas.
Elaboración propia.

Figura 43: Comparación entre muñecas.
Elaboración propia.

Se sustituyeron los tubos de teflón por otros con mayor flexibilidad y con diámetro más pequeño. Se colocó un tubo por movimiento (flexión y extensión) reduciendo la fricción entre hilos y evitando que se enrollen entre sí mismos.

7.6. Yemas de los dedos

Para fabricar la yema de los dedos de la mano animatrónica se tomo como referencia las yemas modeladas de la fase anterior por Omar Galvez. Una de ellas se puede observar en la Figura 44 la cual corresponde a la del dedo pulgar.

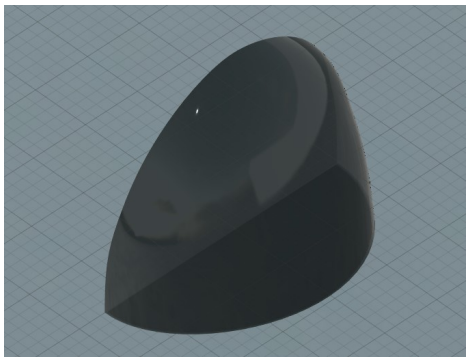


Figura 44: Yema del dedo pulgar [4].

Luego se modeló y fabricó un molde utilizando las yemas anteriormente descritas, pero esta vez tomando en cuenta el espacio donde se unen los hilos que se usan como tendones y el sistema de sujeción. El molde se puede observar en la Figura 45.

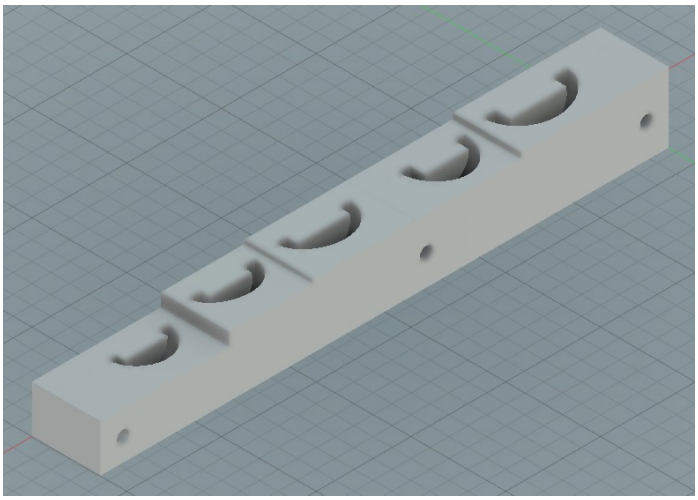


Figura 45: Molde para las yemas.
Elaboración propia.

Se probó utilizar silicona escolar frío, el resultado no fue bueno ya no se seco por completo y se formaron burbujas. En la segunda prueba se utilizó el método de colada de resina el cual corresponde al material caucho de silicona platino el cual es un material para fabricar moldes.

Figura 46: Platinum Silicone Rubber.
Elaboración propia.

Para esta segunda prueba se utilizaron los dos componentes A y B, estos se mezclaron uno a uno en volumen para que puedan reaccionar. Se introdujo la mezcla en el molde y se dejó secar por aproximadamente una hora.

Figura 47: Mezcla de componentes y molde.
Elaboración propia.

Se extrajo la yemas del molde y se recortaron los excedentes, luego se probaron en los dedos y el resultado fue el siguiente:

Figura 48: Dedo pulgar y su yema.
Elaboración propia.

Figura 49: Yemas de todos los dedos.
Elaboración propia.

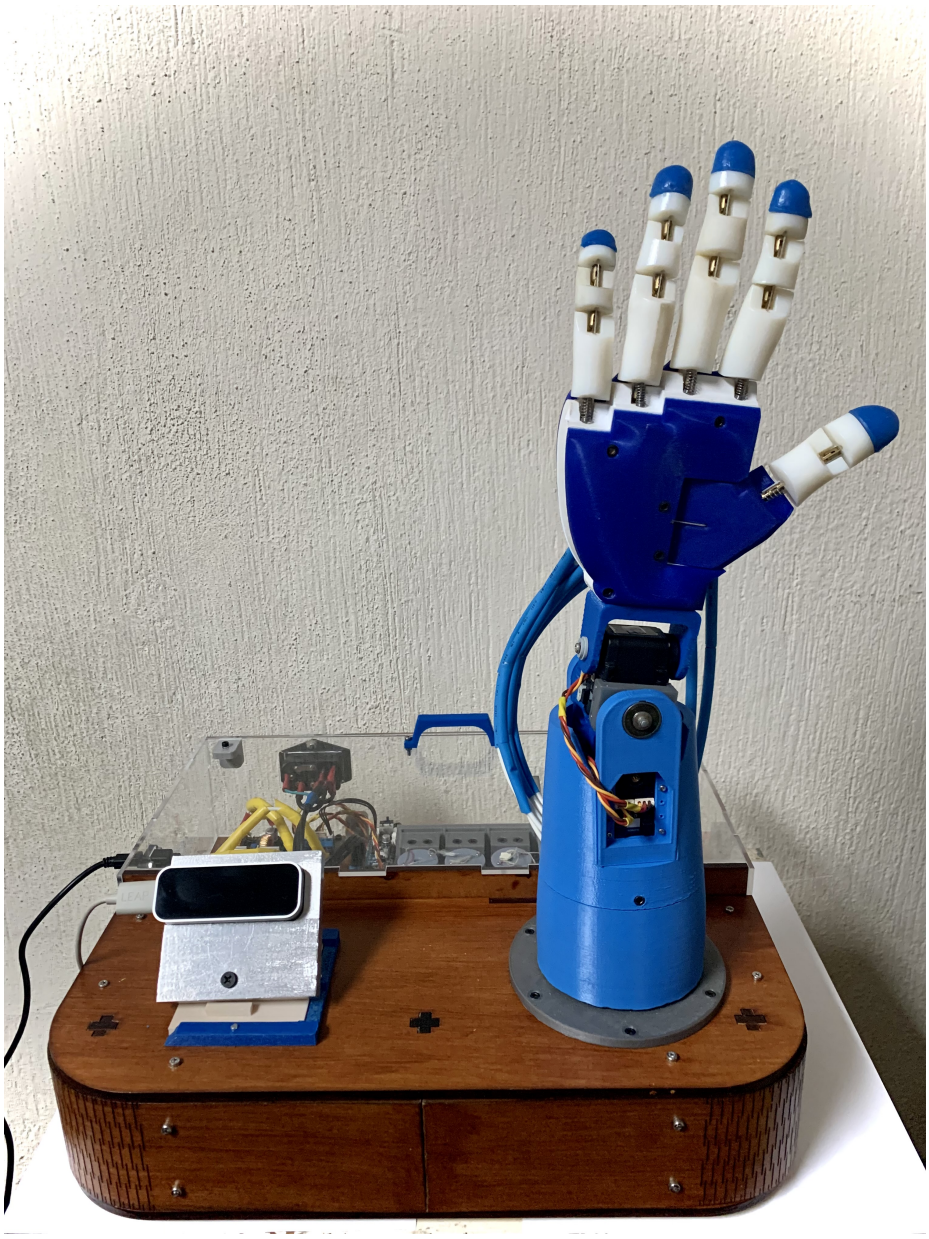


Figura 50: Dedo pulgar y su yema.
Elaboración propia.

CAPÍTULO 8

Interfaz gráfica

Se realizó el siguiente trade study para seleccionar el software que se adapte a las necesidades de este proyecto tomando en cuenta distintos parámetros y a cada uno se le dio un punteo de 0 a 100:

Cuadro 2: Programas con punteos de 0 a 100.

Alternative	Gráficos	Documentación	Codificación	Funciones	Rendimiento	Compatibilidad	Exportar
Unreal	80	70	70	60	80	60	90
Unity	90	100	80	70	80	90	90
LeapC	80	90	80	80	90	90	80
Qt	70	60	60	60	70	70	80
MATLAB	60	80	80	90	60	70	70
ECLIPSE	50	60	60	70	50	60	60

Elaboración propia.

Luego se le asignó un peso a cada criterio dependiendo de la importancia de cada uno:

Cuadro 3: Pesos de cada criterio.

Criteria	Weight	Normalized
Gráficos	20.000	20.000
Documentación	20.000	20.000
Codificación	10.000	10.000
Funciones	10.000	10.000
Rendimiento	15.000	15.000
Compatibilidad	15.000	15.000
Exportar	10.000	10.000

Elaboración propia.

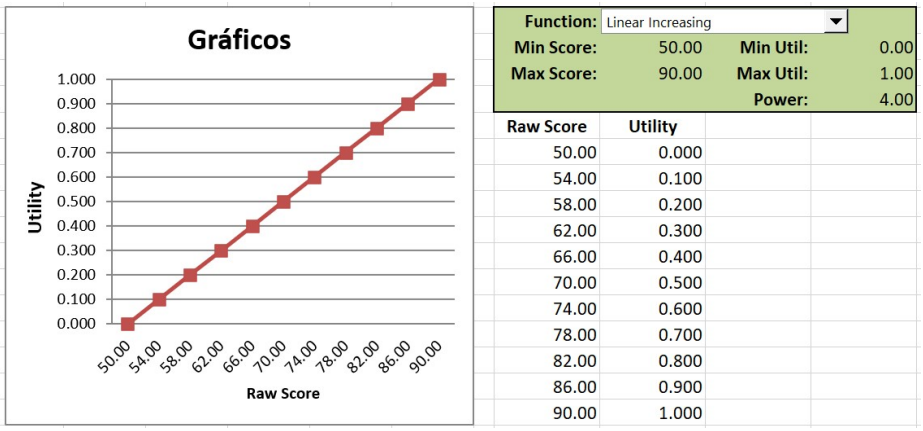


Figura 51: Función utilizada.
Elaboración propia.

Se utilizó la función de incremento lineal para cada criterio, un ejemplo de esto es el siguiente donde se muestra el criterio de gráficos en donde nos muestra la gráfica los valores mínimos y máximos de cada programa y la función utilizada.

Se evaluó cada criterio dependiendo del peso que se le asignó utilizando la función de incremento lineal, el resultado fue el siguiente:

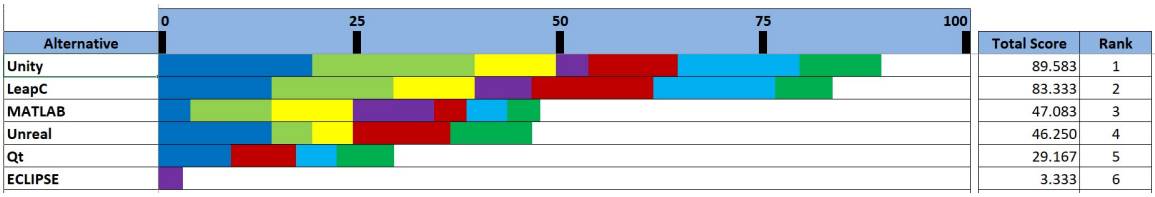


Figura 52: Resultados del Trade Study.
Elaboración propia.

Podemos notar en la Figura 52 que el programa que obtuvo mejor punteo es Unity. Al ser dicho programa el ganador se realizaran algunas pruebas que a continuación se describirán:

Se realizaron pruebas para comprobar el funcionamiento de cada parte del proyecto, como primer paso se probó el correcto funcionamiento de el LEAP Motion con el programa seleccionado.

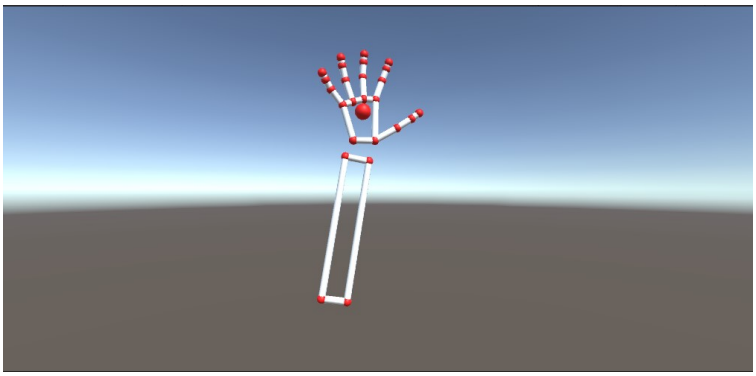


Figura 53: Prueba LEAP Motion - Unity.
Elaboración propia.

Como siguiente paso se probó abrir y cerrar un puerto serial, abriendo el puerto automáticamente al iniciar y cerrarlo con un botón. Se agregó un botón para enviar un texto hacia el microcontrolador.

Figura 54: Prueba Serial - Unity.
Elaboración propia.

Ya que al conectar el microcontrolador a la computadora no siempre es reconocido con la misma dirección se decidió agregar un listado de puertos disponibles, un botón para actualizar los puertos, un botón para conectar y cerrar puerto y un puerto para enviar datos al microcontrolador.

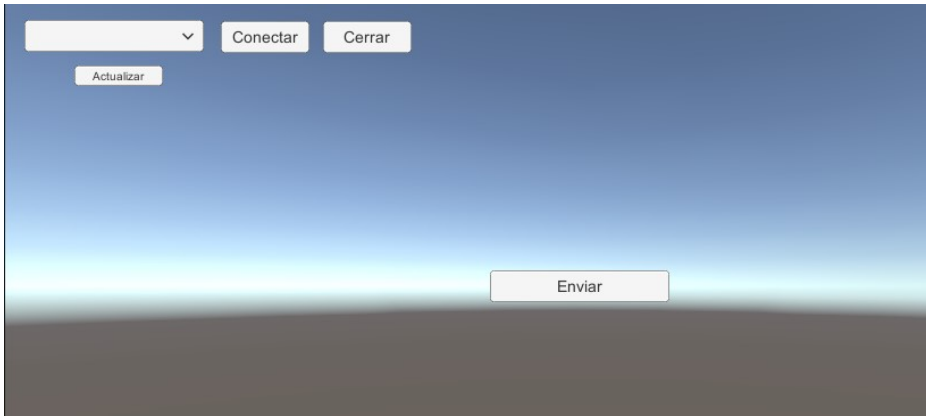


Figura 55: Prueba Serial - Unity - Puertos.
Elaboración propia.

Por último se agregó el Leap al prototipo y se comenzó a obtener posiciones de la mano y se mandaron en un arreglo de bytes al microcontrolador.

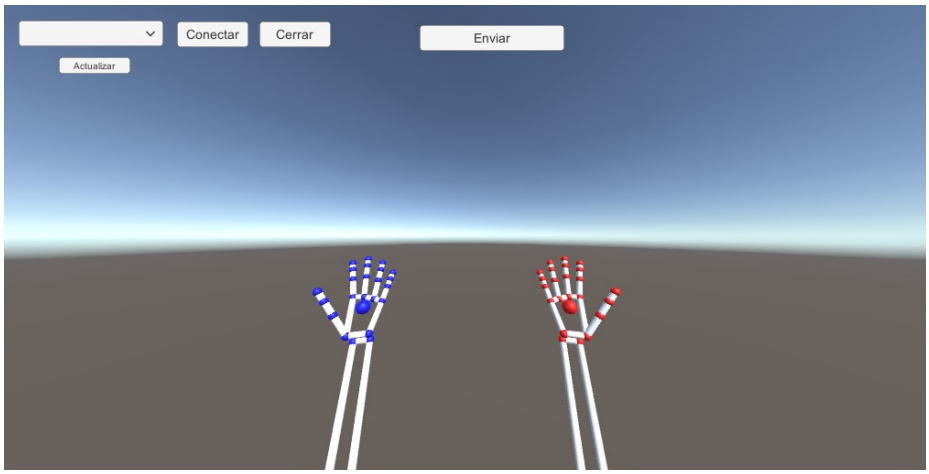


Figura 56: Prueba Unity - Leap.
Elaboración propia.

8.1. Diseño

Se propuso un diseño en el que se mostrará la lectura de mano y se replique en el modelo animatrónico, la mano se muestra en diferentes tipos de modelos y estos se pueden ir cambiando con las flechas del teclado (se dejo un pequeño texto con las instrucciones para esto).

Se agregó un botón para seleccionar el puerto USB al que queremos conectar y se agregaron los botones de conectar y cerrar. Si el puerto se desconecta se agregó un mensaje de .ERROR DE PUERTO"para notificar al usuario que este no está disponible. Para quitar el mensaje se debe de presionar el botón de actualizar.

Al momento de tener conectado el puerto USB se habilitan las opciones de enviar para empezar a sincronizar el movimiento de la interfaz con el de la mano animatrónica. Si se desea dejar de enviar datos se debe de presionar el botón de parar envio y con este se habilitan los botones de cerrar puerto y de volver a enviar.

Figura 57: Interfaz propuesta para controlar la mano.
Elaboración propia.

8.2. Scripts

Se utilizaron dos scripts para procesar datos. En el script llamado Serial se crearon las funciones relacionadas con la conexión al puerto USB y de habilitación de botones y en el script llamado EnvioDatos se crearon las funciones que detectan y procesan los ángulos útiles para el control de la mano. Se relacionaron entre sí con un game object para poder utilizar funciones entre scripts.

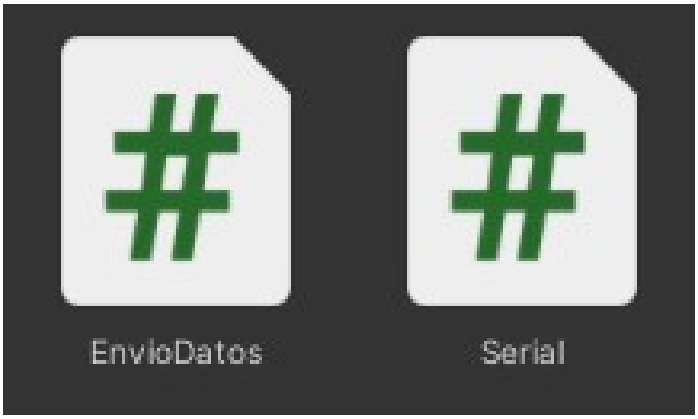


Figura 58: Scripts utilizados.
Elaboración propia.

8.3. Funciones

En el script llamado Serial se agregaron funciones para conectar un puerto serial a 9600 baudrates utilizando un try y catch con el objtivo de que la aplicación no falle al conectar un puerto no disponible, se creo una función para actualizar los puertos desplegándolos en la interfaz, una función para cerrar el puerto, una función para escribir en el puerto USB un array de 10 bytes y una función propia de Unity llamada OnApplicationQuit que si el puerto esta conectado al cerrar la aplicación desconecte el puerto con el objetivo de que no quede bloqueado.

En el script de EnviarDatos se creó una función para leer y procesar el frame, se utilizó la programación orientada a objetos propuesta por LeapMotion, utilizando quaterniones, vectores, listas, Finger, Bone, Hand y conversiones de variables. Se creó una función para convertir quaterniones a ángulos de Euler ya que la función propia de Unity no lo hacía en el orden correcto de ejes que se necesitaba. Se creó una función para acotar los datos en sus ángulos máximos y mínimos antes de ser enviados por Serial y otra función para limitar los ángulos de lectura del sensor.

CAPÍTULO 9

Análisis de elementos finitos

Para realizar este análisis se utilizó el software de Autodesk Inventor con el objetivo de obtener esfuerzos en puntos críticos del diseño en las poses que más esfuerzos generan. Al ser este proyecto de exhibición, no soportará cargas externas.

Los esfuerzos que se encuentran en el diseño son generados por el propio peso de los componentes y estos recaen sobre el diseño de la muñeca. Para este análisis los esfuerzos son generados por la gravedad y esta se tomó como $9.807 m/s^2$.

Se realizó un análisis estático en tres posiciones combinadas que son críticas:

- Desviación cubital

- Extensión y desviación cubital

- Flexión y desviación cubital

No se realizó análisis en el movimiento de desviación radial ya que es un movimiento muy limitado. El movimiento de desviación cubital es un poco más amplio y genera mayor torque a la muñeca por recargar más peso sobre ella.

Al realizar el análisis se ebtuvieron los siguientes resultados:

Cuadro 4: Resultado del análisis de elementos finitos.

Prueba	Resultado	Movimiento	Esfuerzo máximo	Desplazamiento máximo
Primera	Figura 60	Desviación cubital	4.005 MPa	0.003468 mm
Segunda	Figura 61	Extensión y desviación cubital	27.83 MPa	0.02101 mm
Tercera	Figura 62	Flexión y desviación cubital	36.67 MPa	0.03104 mm

Elaboración propia.

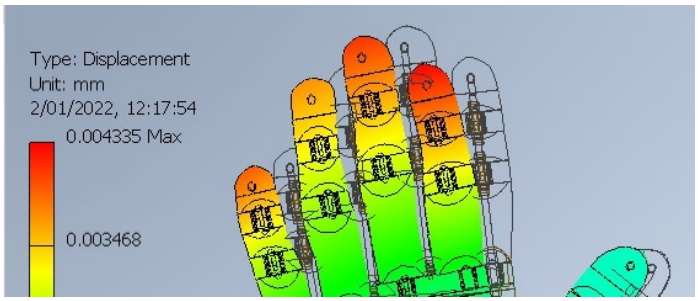
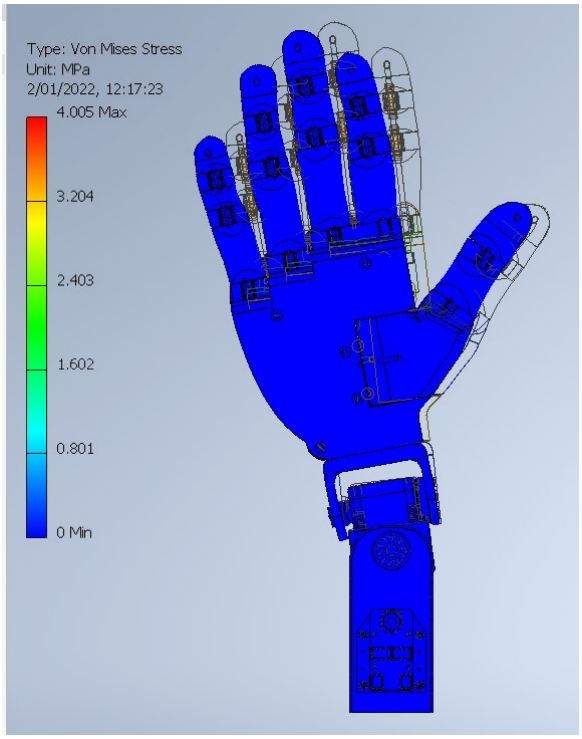


Figura 59: Ejemplo de análisis.
Elaboración propia.

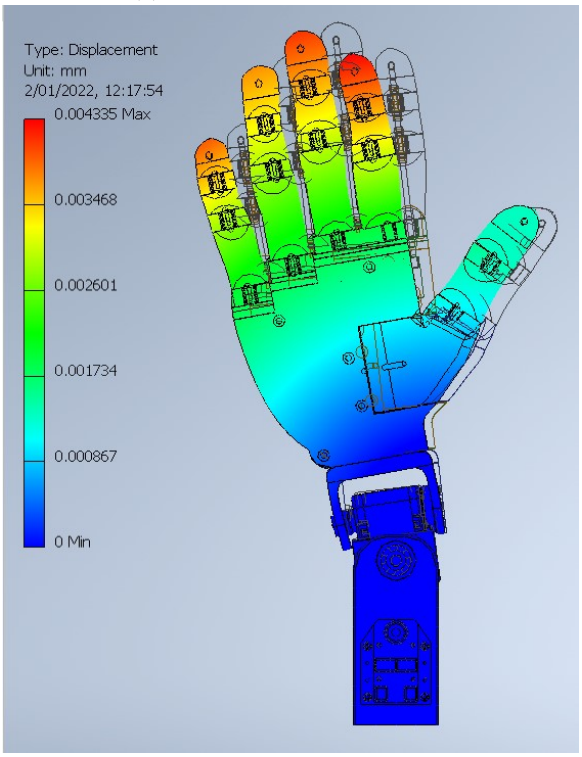
Al analizar los resultados se observa que el desplazamiento máximo está presente en todos los dedos, pero este está en milésimas de milímetro por lo que no tiene relevancia en el diseño.

Las estructuras de soporte de la muñeca fueron fabricadas en PET, por lo que al comparar su límite elástico (72 MPa) con los resultados obtenidos en el análisis y estos no superan el límite. Con respecto al factor de suguridad (límite elástico dividido esfuerzo máximo) podemos notar que es mayor a uno por lo que las piezas se mantendrán en la zoma elástica y el diseño es válido.

Otro factor a tomar en cuenta es que las piezas no están completamente rellenas ya que se fabricaron entre un $15-20\,\%$ de relleno y el análisis las toma como si fueran completamente rellenas. Esto significa que los esfuerzos son mucho menores a los obtenidos ya que disminuye las fuerzas por gravedad al reducir el peso de las piezas.

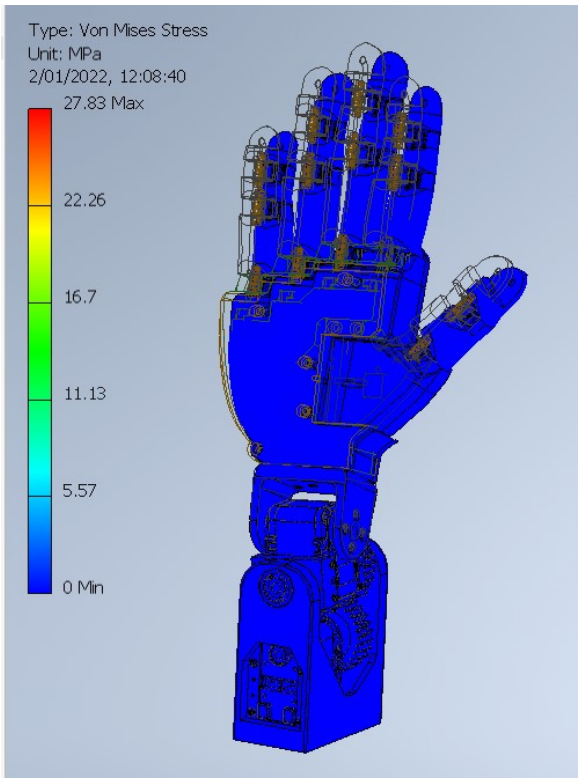


(a) Análisis de esfuerzo máximo

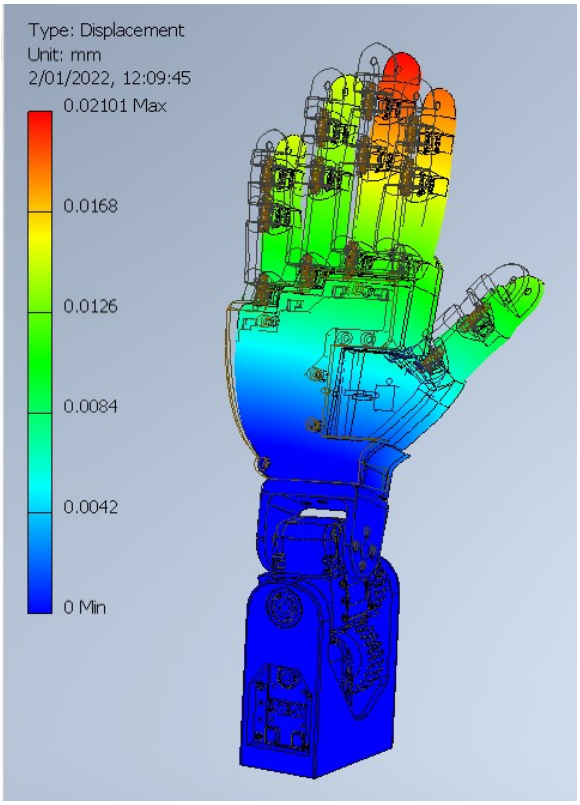


(b) Análisis de desplazamiento máximo

Figura 60: Análisis en desviación cubital.
Elaboración propia.

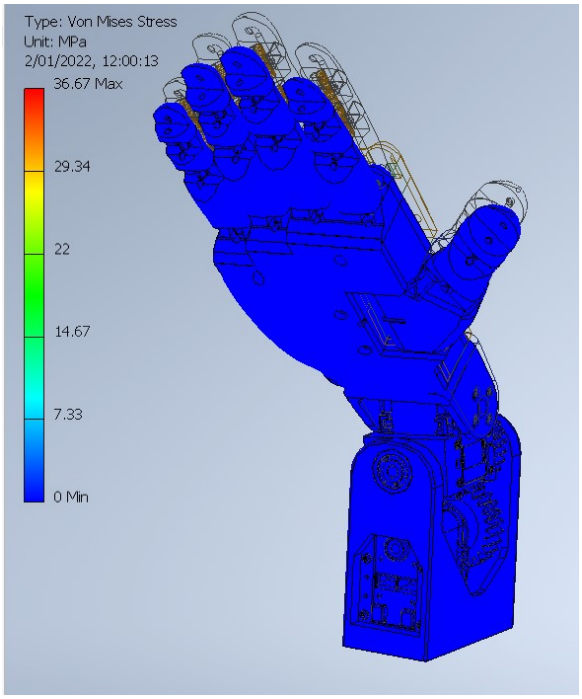


(a) Análisis de esfuerzo máximo

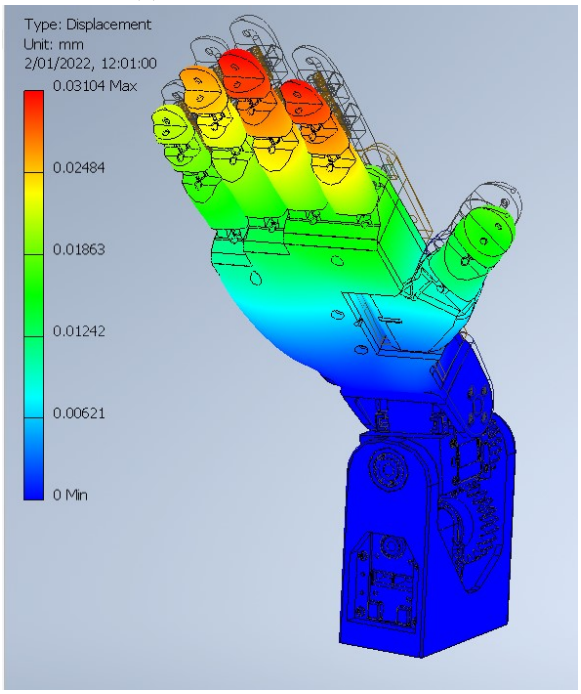


(b) Análisis de desplazamiento máximo

Figura 61: Análisis en extensión y desviación cubital.
Elaboración propia.



(a) Análisis de esfuerzo máximo



(b) Análisis de desplazamiento máximo

Figura 62: Análisis en extensión y desviación cubital. Elaboración propia.

CAPÍTULO 10

Conclusiones

- Se solucionó el problema de unión entre hilos en las falanges distales de los dedos utilizando un clavo pequeño.
- Se diseñaron y fabricaron los dedos reduciendo la movilidad y modificando los conductos para mejorar el movimiento de estos.
- Se logró el movimiento del dedo pulgar cambiando de posición el tubo de teflón al metacarpo.
- Se modificó la estructura de la palma para mejorar la unión de los tubos de teflón y el trayecto de estos.
- Se implementó una interfaz gráfica de usuario en Unity para mejorar la interacción del prototipo con el usuario.
- Se logró reducir la temperatura en largos tiempos de uso con una base de aluminio para el Leap Motion.
- Se rediseñó la base del proyecto con el objetivo de incorporar el Leap Motion y mostrar todos los componentes.
- Se fabricaron las yemas de los dedos utilizando un material flexible.
- Se rediseñó la muñeca y el antebrazo mejorando la apariencia de la mano.
- Se reemplazaron los tubos de teflón por unos de mayor longitud mejorando el movimiento de la muñeca.
- Se realizó un análisis de elementos finitos con posiciones críticas de la mano.
- Se validó el diseño con el análisis de elementos finitos.

CAPÍTULO 11

Recomendaciones

- Utilizar el controlador versión 5 Gemini del Leap Motion (si ya se encuentra disponible) y actualizar la programación ya que con esto mejora la detección de la mano en la mayoría de posiciones.
- Fabricar la mano izquierda para utilizar la detección de ambas manos del sensor.
- Agregar el controlador para la mano izquierda en la programación de la interfaz gráfica de usuario.
- Fabricar los engranajes con aluminio y con mayor número de dientes para mejorar el movimiento y resistencia de la muñeca.
- Agregarle más grados de libertad a la mano para mejorar el prototipo.

CAPÍTULO 12

Bibliografía

[1] R. Murray, "A Mathematical Introduction to Robotic Manipulation," *CRC Press*, 2017.

[2] P. Mazariegos, "Diseño e implementación de un nuevo modelo de la mano de la Prótesis Biónica Transhumeral," *Universidad del Valle de Guatemala*, 2012.

[3] B. Rodas, "Diseño e implementación de mano y antebrazo animatrónico antropomorfo," *Universidad del Valle de Guatemala*, 2019.

[4] O. Gálvez, "Optimización de diseño de una mano y muñeca animatrónica antropomórfica de la fase tres e implementación de un control interactivo," *Universidad del Valle de Guatemala*, 2020.

[5] M. Romero, A. S. an C. San Miguel, L. Sáez y P. González, *Prototipo de mano robótica antropométrica sub-actuada*. 2012, págs. 45-59.

[6] RAE, *Tendón*. dirección: https://dle.rae.es/.

[7] J. Doyle y M. Botte, "Surgical Anatomy of the Hand and Upper Extremity," *Lippincott Williams & Wilkins*, 2003.

[8] J. Bustamante y M. J. Saldivia, *Atlas de Osteología Humana por tomografía computacional tridimencional*. Editorial de la Universidad de la Plata, 2018, págs. 53-56.

[9] P. Ruiz y C. Cevallos, *Estudio cinemático de la mano*. dirección: https://www.ecorfan.org/booklets/Booklets_CIERMMI_2019/.

[10] musculos.org, *Músculos de la mano*. dirección: https://www.musculos.org/.

[11] C. Sanz y E. Cepero, "El pulgar, Cinesiología," *Universidad de Zaragoza*, 2020.

[12] R. T. D. S. E. Cuba, *Biomecánica del complejo articular del codo*, feb. de 1999. dirección: http://www.sld.cu/sitios/rehabilitacion-bio/temas.php.

[13] T. Angulo, A. Álvarez e Y. Fuentes, *Biomecánica*. dirección: http://www.revistareduca.es/index.php/reduca-enfermeria/article/viewFile/752/768.

[14] PNTIC, *¿Qué significa CAD, CAM y CAE?* Dirección: http://platea.pntic.mec.es/~jalons3/4ESO/1diseno/1cacaca.htm.

[15] IAC, *Autodesk Fusion 360*. dirección: https://www.iac.com.co/fusion-360/.

[16] ultraleap, *LEAP Motion Controller*. dirección: https://www.ultraleap.com/product/leap-motion-controller/.

[17] P. Lázaro, "Módulo de reconocimiento gestual para control de robot en tareas de asistencia II," *Universidad Carlos III de Madrid*, 2017.

[18] U. Technologies, *Unity*. dirección: https://unity.com/es.

[19] E. Games, *Unreal Engine*. dirección: https://www.unrealengine.com/en-US/.

[20] ultraleap, *LeapC API*. dirección: https://developer.leapmotion.com/documentation/v4/index.html.

[21] Q. Company, *QT Framework*. dirección: https://www.qt.io/.

CAPÍTULO 13

Anexos

13.1. Código para leer el Leap Motion con Unity

```
   // Universidad del Valle de Guatemala
// Dise o e innovaci n 2
// Mano animatr nica

// Miguel Alejandro Garc a de Le n
// 17560
// Ingenier a Mecatr nica

// Se tomo como referencia parte del c digo de Omar G lvez

// Incluimos librer as de Unity
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Leap;
using Leap.Unity;
using LeapInternal;
using System.IO.Ports;
using System;
using UnityEngine.UI;

public class EnvioDatos : MonoBehaviour
{
    // Conectamos con el otro script
    public Serial serial;

    // Definimos el controlador
    Controller controller;

    // Definimos los  ngulos  m ximos y m nimos para cada grado de
    libertad
    const int MAX_ANTE = 60; const int MIN_ANTE = -60;
    const int MAX_MUNE_DES = 30; const int MIN_MUNE_DES = -15;
    const int MAX_MUNE_EXT = 30; const int MIN_MUNE_EXT = -30;
```

```
34      const int MAX_INDI = 90; const int MIN_INDI = 0;
35      const int MAX_MEDI = 90; const int MIN_MEDI = 0;
36      const int MAX_ANUL = 90; const int MIN_ANUL = 0;
37      const int MAX_MENI = 90; const int MIN_MENI = 0;
38      const int MAX_PULG = 110; const int MIN_PULG = -10;
39      const int MAX_PULG_META = 70; const int MIN_PULG_META = -10;
40
41      public float ang_antebrazo_trans;
42
43      void Start()
44      {
45          // Controlador
46          controller = new Controller();
47
48          // Leemos y enviamos datos despu s del primer segundo y luego en
    periodos de 0.5 segundos
49          InvokeRepeating("LeapUnity", 1.0f, 0.05f);
50      }
51
52      // Update is called once per frame
53      void Update()
54      {
55
56      }
57
58      public void LeapUnity() {
59
60          // Leemos el frame del controlador
61          Frame frame = controller.Frame();
62
63          // Inicializamos variables
64          float ang_pulg_meta;
65          float ang_menique;
66          float ang_anular;
67          float ang_medio;
68          float ang_indice;
69          float ang_pulgar;
70
71          // Si se detecta m s de una mano continuamos con la operaci n
72          if (frame.Hands.Count > 0)
73          {
74              // Listamos la cantidad de manos
75              List<Hand> hands = frame.Hands;
76
77              // Leemos la primer mano detectada
78              Hand hand = hands [0];
79
80              // Definimos esa mano como derecha y comprobamos
81              bool derecha = hand.IsRight;
82              if (derecha)
83              {
84                  //Debug.Log("Es una mano derecha");
85
86                  // ----------- PRONACI N - SUPINACI N -----------------
87                  LeapQuaternion antebrazo = hand.Arm.Rotation;           //
    LeapQuaternion -> Quaternion -> Vector3 -> Vector3 euler
88                  Quaternion antebrazoQt = antebrazo.ToQuaternion();
89                  Vector3 antebrazoEu = Euler_from_quaternion(antebrazoQt);
90                  Vector3 antebrazoEuD = antebrazoEu * Mathf.Rad2Deg;    //
```

```
    Vector3 radianes -> Vector3 degree
91
92              double ang_antebrazo_direccion = antebrazoEuD.y;
93
94              // ----EXTENSION Y FLEXION ---- DESVIACION RADIAL Y CUBITAL
    ------
95
96              double ang_pal_x = 90 - Math.Acos(hand.Direction.y) * Mathf.
    Rad2Deg;
97              double ang_pal_y = -(90 + Math.Atan2(hand.Direction.z, hand.
    Direction.x) * Mathf.Rad2Deg);
98
99              double ang_mun_ext = antebrazoEu.x - ang_pal_x;
100             double ang_mun_des = ang_antebrazo_direccion - ang_pal_y;
101
102             //
    ----------------------------------------------------------------
103
104             // Leemos el vector normal de la palma
105             Vector vec_normal = hand.PalmNormal;
106
107             // Definimos una lista de dedos
108             List<Finger> fingers = hand.Fingers;
109
110             // ------------------- METACARPO DEL PULGAR
    ------------------------
111
112             Finger dedo_pulg = fingers[0];
113             Leap.Bone meta_pulg = dedo_pulg.Bone(Bone.BoneType.
    TYPE_INTERMEDIATE);
114             Vector pulg_meta_p = meta_pulg.PrevJoint;
115             Vector pulg_meta_n = meta_pulg.NextJoint;
116             Vector pulg_meta_vect = pulg_meta_n - pulg_meta_p;
117             ang_pulg_meta = -3 * Ang_prod_pun(vec_normal, pulg_meta_vect
    ) + 230;
118             ang_pulg_meta = Ang_limit(ang_pulg_meta, 70, 0);
119             Debug.Log("pulgar   " + ang_pulg_meta);
120
121             // ------------------------- MENIQUE
    ------------------------------
122
123             Finger dedo_meni = fingers[4];
124             Leap.Bone menique = dedo_meni.Bone(Bone.BoneType.
    TYPE_INTERMEDIATE);
125             Vector menique_p = menique.PrevJoint;
126             Vector menique_n = menique.NextJoint;
127             Vector menique_vect = menique_n - menique_p;
128             ang_menique = -3 * Ang_prod_pun(vec_normal, menique_vect) +
    230;
129
130             ang_menique = Ang_limit(ang_menique, 100, 0);
131
132             // ------------------------- ANULAR
    ------------------------------
133
134             Finger dedo_anu = fingers[3];
135             Leap.Bone anular = dedo_anu.Bone(Bone.BoneType.
    TYPE_INTERMEDIATE);
136             Vector anular_p = anular.PrevJoint;
```

```
137                 Vector anular_n = anular.NextJoint;
138                 Vector anular_vect = anular_n - anular_p;
139                 ang_anular = -3 * Ang_prod_pun(vec_normal, anular_vect) +
    230;
140 
141                 ang_anular = Ang_limit(ang_anular, 100, 0);
142 
143                 // ------------------------- MEDIO
    ----------------------------
144 
145                 Finger dedo_med = fingers[2];
146                 Leap.Bone medio = dedo_med.Bone(Bone.BoneType.
    TYPE_INTERMEDIATE);
147                 Vector medio_p = medio.PrevJoint;
148                 Vector medio_n = medio.NextJoint;
149                 Vector medio_vect = medio_n - medio_p;
150                 ang_medio = -3 * Ang_prod_pun(vec_normal, medio_vect) + 230;
151 
152                 ang_medio = Ang_limit(ang_medio, 100, 0);
153 
154                 // ------------------------- INDICE
    ----------------------------
155 
156                 Finger dedo_ind = fingers[1];
157                 Leap.Bone indice = dedo_ind.Bone(Bone.BoneType.
    TYPE_INTERMEDIATE);
158                 Vector indice_p = indice.PrevJoint;
159                 Vector indice_n = indice.NextJoint;
160                 Vector indice_vect = indice_n - indice_p;
161                 ang_indice = -3 * Ang_prod_pun(vec_normal, indice_vect) +
    230;
162 
163                 ang_indice = Ang_limit(ang_indice, 100, 0);
164 
165                 // ------------------------- PULGAR
    ----------------------------
166 
167                 Leap.Bone dedo_pulg_2 = dedo_pulg.Bone(Bone.BoneType.
    TYPE_PROXIMAL);
168                 Vector pulgar_2_p = dedo_pulg_2.PrevJoint;
169                 Vector pulgar_2_n = dedo_pulg_2.NextJoint;
170                 Vector pulgar_2_vect = pulgar_2_n - pulgar_2_p;
171 
172                 Leap.Bone dedo_pulg_3 = dedo_pulg.Bone(Bone.BoneType.
    TYPE_DISTAL);
173                 Vector pulgar_3_p = dedo_pulg_3.PrevJoint;
174                 Vector pulgar_3_n = dedo_pulg_3.NextJoint;
175                 Vector pulgar_3_vect = pulgar_3_n - pulgar_3_p;
176 
177                 ang_pulgar = 2 * Ang_prod_pun(pulgar_2_vect, pulgar_3_vect)
    - 25;
178 
179                 ang_pulgar = Ang_limit(ang_pulgar, 110, -10);
180 
181                 //---------------------- ARREGLO DE DATOS
    ----------------------
182                 // Acotamos los datos para enviarlos por serial
183 
184                 int ante = Cota(-antebrazoEuD.z, MAX_ANTE, MIN_ANTE);
```

```
185                 int mun_des = Cota(ang_mun_des, MAX_MUNE_DES, MIN_MUNE_DES);
186                 int mun_ext = Cota(ang_mun_ext, MAX_MUNE_EXT, MIN_MUNE_EXT);
187                 int pulg = Cota(ang_pulgar, MAX_PULG, MIN_PULG);
188                 int pulg_meta = Cota(ang_pulg_meta, MAX_PULG_META,
    MIN_PULG_META);
189                 int ind = Cota(ang_indice, MAX_INDI, MIN_INDI);
190                 int med = Cota(ang_medio, MAX_MEDI, MIN_MEDI);
191                 int anul = Cota(ang_anular, MAX_ANUL, MIN_ANUL);
192                 int meni = Cota(ang_menique, MAX_MENI, MIN_MENI);
193
194                 //---------------------- ENVIO DE DATOS
    ----------------------
195
196                 if (serial.isConnected)
197                 {
198                     //Debug.Log("---------Enviando---------" + antebrazoEuD.
    z + ante);(byte) mun_des
199
200                     // Codigo para probar unicamente un grado de libertad
201                     // Descomentar el que se quiere probar y comentar el
    completo
202
203                     //byte[] enviar = {255, (byte)ante, 127, 127, 127, 127,
    127, 127, 127, 127};
204                     //byte[] enviar = { 255, 127, (byte)mun_des, 127, 127,
    127, 127, 127, 127, 127 };
205                     //byte[] enviar = { 255, 127, 127, (byte)mun_ext, 127,
    127, 127, 127, 127, 127 };
206                     //byte[] enviar = { 255, 127, 157, 127, (byte)pulg, (
    byte)pulg_meta, 127, 127, 127, 127 };
207                     //byte[] enviar = { 255, 127, 127, 127, 127, (byte)
    pulg_meta, 127, 127, 127, 127 };
208                     //byte[] enviar = { 255, 127, 127, 127, 127, 127, (byte)
    ind, 127, 127, 127 };
209                     //byte[] enviar = { 255, 127, 127, 127, 127, 127, 127, (
    byte)med, 127, 127 };
210                     //byte[] enviar = { 255, 127, 127, 127, 127, 127, 127,
    127, (byte)anul, 127 };
211                     //byte[] enviar = { 255, 127, 127, 127, 127, 127, 127,
    127, 127, (byte)meni };
212
213
214                     // Arreglamos los datos convirtiendolos a un arreglo de
    bytes
215                     byte[] enviar = { 255, (byte)ante, (byte)mun_des, (byte)
    mun_ext, (byte)pulg, (byte)pulg_meta, (byte)ind, (byte)med, (byte)anul,
    (byte)meni };
216
217                     // Enviamos los datos por serial
218                     serial.SendMessageToArduino(enviar);
219
220                 }
221             }
222         }
223     }
224
225     // Funci n para convertir quaternion a  ngulos  de euler en XYZ
226     public Vector3 Euler_from_quaternion(Quaternion q)
227     {
```

```
228         double t0 = 2.0 * (q.w * q.x + q.y * q.z);
229         double t1 = 1.0 - 2.0 * (q.x * q.x + q.y * q.y);
230         double roll_x = Math.Atan2(t0, t1);
231 
232         double t2 = 2.0 * (q.w * q.y - q.z * q.x);
233         if (t2 > 1.0)
234         {
235             t2 = 1.0;
236         }
237         else if (t2 < -1.0)
238         {
239             t2 = -1.0;
240         }
241         else
242         {
243             t2 = 1.0 * t2;
244         }
245         double pitch_y = Math.Asin(t2);
246 
247         double t3 = 2.0 * (q.w * q.z + q.x * q.y);
248         double t4 = 1.0 - 2.0 * (q.y * q.y + q.z * q.z);
249         double yaw_z = Math.Atan2(t3, t4);
250 
251         return new Vector3((float)roll_x, (float)pitch_y, (float)yaw_z);
252     }
253 
254     // Funci n para acotar los datos
255     public int Cota(double valor, int max, int min)
256     {
257         double acotado = 127 + valor;
258         if (valor > max)
259         {
260             acotado = 127 + max;
261         }
262         else if (valor < min)
263         {
264             acotado = 127 + min;
265         }
266         acotado = Math.Round(acotado);
267 
268         return (int)acotado;
269     }
270 
271     // Funci n para calcular el producto punto y magnitud, devuelve el
    arcos en degree
272     public float Ang_prod_pun(Vector vec1, Vector vec2){
273 
274         float dato1 = vec1.x* vec2.x + vec1.y * vec2.y + vec1.z* vec2.z;
275         float dato2 = vec1.Magnitude * vec2.Magnitude;
276 
277         float ang = (float)Math.Acos(dato1/dato2) * Mathf.Rad2Deg;
278 
279         return ang;
280     }
281 
282     // Funci n para limitar el  ngulo  m ximo de la lectura de datos
283     public float Ang_limit(float angulo,float max, float min)
284     {
285         if (angulo > max)
```

```
        {
            angulo = max;
        }
        else if (angulo < min)
        {
            angulo = min;
        }

        return angulo;
    }

}
```

13.2. Código para utilizar el puerto serial y los botones

```
   // Universidad del Valle de Guatemala
// Dise o e innovaci n 2
// Mano animatr nica

// Miguel Alejandro Garc a de Le n
// 17560
// Ingenier a Mecatr nica


// Incluimos librer as de Unity
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO.Ports;
using System;
using UnityEngine.UI;

public class Serial : MonoBehaviour
{
    // Definimos el puerto serial
    public SerialPort myPort;

    // Definimos los enlaces para los botones
    [SerializeField] Dropdown myDrop;
    [SerializeField] Button conectarBtn;
    [SerializeField] Button desconectarBtn;
    [SerializeField] Button enviarBtn;
    [SerializeField] Button pararBtn;
    [SerializeField] Text errorText;
    //[SerializeField] GameObject serialCanvas;

    // Variables
    public bool isConnected = false;

    void Start() // Funci n inicial
    {
        RefreshPorts();
        isConnected = false;
        desconectarBtn.gameObject.SetActive(false);
        pararBtn.gameObject.SetActive(false);
        enviarBtn.gameObject.SetActive(false);
        errorText.gameObject.SetActive(false);
```

```
43      }
44
45      public void SendMessageToArduino(byte[] msg) // Funci n para enviar un
    arreglo de 10 bytes por el puerto serial
46      {
47          myPort.Write(msg, 0, 10);
48          //Debug.Log(msg);
49      }
50
51      public void PararBtn() // Funci n para detener el env o de datos por
    serial
52      {
53          isConnected = false;
54          pararBtn.gameObject.SetActive(false);
55          desconectarBtn.gameObject.SetActive(true);
56          enviarBtn.gameObject.SetActive(true);
57      }
58
59      public void EnviarBtn() // Funci n para env ar datos por serial
    habilitando la variable "isConnected"
60      {
61          isConnected = true;
62          Debug.Log("---------Enviando---------");
63          enviarBtn.gameObject.SetActive(false);
64          pararBtn.gameObject.SetActive(true);
65          desconectarBtn.gameObject.SetActive(false);
66      }
67
68      public void ClosePort() // Funci n para cerrar el puerto serial y
    enviar posiciones iniciales a los servos
69      {
70          byte[] enviar = { 255, 127, 127, 127, 127, 127, 127, 127, 127, 127
    };
71          SendMessageToArduino(enviar);
72          myPort.Close();
73          conectarBtn.gameObject.SetActive(true);
74          desconectarBtn.gameObject.SetActive(false);
75          enviarBtn.gameObject.SetActive(false);
76      }
77
78      public void RefreshPorts() // Funci n para actualizar los puertos
    seriales conectados al equipo
79      {
80          List<string> ports = new List<string> { };
81          foreach (string port in SerialPort.GetPortNames())
82          {
83              ports.Add(port);
84          }
85          myDrop.ClearOptions();
86          //myDrop.captionText.fontSize = 12;
87          myDrop.AddOptions(ports);
88          errorText.gameObject.SetActive(false);
89      }
90
91      public void AttemptConnection() // Funci n para conectar un puerto
    serial seleccionado en el Drop
92      {
93          myPort = new SerialPort(myDrop.captionText.text, 9600); // Conexi n
     del puerto
```

```
94          try
95          {
96              myPort.Open();
97              myPort.ReadTimeout = 10;
98              errorText.gameObject.SetActive(false);
99              conectarBtn.gameObject.SetActive(false);
100             enviarBtn.gameObject.SetActive(true);
101             //desconectarBtn.gameObject.SetActive(true);
102             Debug.Log("---------Conectado---------");
103         }
104         catch (System.Exception ex) // Si no se conecta el puerto activar
    un mensaje de error
105         {
106             errorText.gameObject.SetActive(true);
107             errorText.text = ex.ToString().Substring(0, ex.ToString().Length
     / 4);
108         }
109     }
110
111     void OnApplicationQuit() // Funci n para cerrar el puerto serial si la
    aplicaci n se cierra
112     {
113         if (isConnected)
114         {
115             byte[] enviar = { 255, 127, 127, 127, 127, 127, 127, 127, 127,
    127 };
116             SendMessageToArduino(enviar);
117             myPort.Close();
118             conectarBtn.gameObject.SetActive(true);
119             desconectarBtn.gameObject.SetActive(false);
120         }
121         Debug.Log("Application ending after " + Time.time + " seconds");
122     }
123 }
```