UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo e implementación de un modelo cinemático en el embodiment de un sistema HMI de alta exactitud para la mano

TESIS

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Isaac Espinosa Zariñán

DIRECTOR DE TESIS

M.I. César Abraham Luna Estrada

Ciudad Universitaria, Cd. Mx., 2024

AGRADECIMIENTOS

En este espacio quiero abrirme y plasmar mis pensamientos acerca de la vida que tuve, así como de las personas que me acompañaron en la misma.

Desde niño siempre me consideré como alguien tonto, debido a que siempre pude dar más de mí, pero nunca lo hice, incluso a la fecha me siento igual, me abruma ver que en el mundo hay tantas cosas por aprender, y que aún me considero un aprendiz, incluso en las áreas que domino, ya que yo no lo se todo, pero incluso sin saberlo todo, logre llegar alto, no por heredar la inteligencia de mis padres, ni incluso por ir en la UNAM, fue por la disciplina, enseñanza, y apoyo de mi familia.

Quiero agradecer a mi madre ***Magdalena Zariñán García***, una mujer a la cual admiro, porque ella siempre demostró ser una persona inteligente, no tuvo que demostrarlo con palabras, si no con acciones, ella siempre fue mejor que cualquiera, logro sacar a 2 hijos sus carreras universitarias, aguanto en muchos casos mi mal humor, y mis groserías, cuando ella no tenía por qué hacerlo, pero lo hizo, agradezco sus enseñanzas y me siento orgulloso de haber heredado su tenacidad y su forma de pensar: "no dejare que esta problemática me gane", aunque no lo diga nunca. ***TE AMO MAMÁ, GRACIAS POR TODO***.

Quiero agradecer a mi padre ***José Gerardo Espinosa Garibay***, un hombre fuerte, que nunca logro doblegarse ante cualquier circunstancia, alguien que sin importar la perdida, siempre siguió luchando, para darnos gustos y ayudarnos en todo lo que necesitáramos, admiro su fortaleza, y espero en un futuro llegar a ser igual de fuerte, ser alguien que nunca se rinda sin importar los desvelos ni el cansancio, a pesar de ser algo rudo, quiero rescatar su mejor enseñanza que me dio al hacer una tarea mal en la primaria: "yo no estoy educando a un mediocre", una enseñanza que me dio carácter y me hizo ser quien soy. TE AMO PAPA, GRACIAS POR TODO.

Quiero agradecer a mi hermano ***Gerardo Espinosa Zariñán***, eres un hombre el cual, desde niño siempre admire, por su inteligencia y su forma de resolver los problemas del día a día, desde chico siempre me enseñabas tus secretos, incluso cuando los consideraba tontos y ridículos, pero ahora veo, que tengo suerte de tener un hermano el cual siempre me apoyo, incluso cuando yo llegue a ser a veces un mal hermano, a veces te falle

por mi inmadurez, y mi rebeldía, perdón por todo eso, y gracias por siempre creer en mí. TE QUIERO MUCHO TOTOMANITO.

Agradezco a mi asesor ***César Abraham Luna Estrada***, quien me apoyo estos últimos años, no solo en el ámbito académico, también profesional, brindándome su apoyo en los proyectos y dándome la oportunidad de demostrar mis capacidades laborales, ¡Muchas gracias!

Quiero agradecer a mis profesores del bachillerato ***José Luis Perales Rico*** *y* ***Francisco Miguel Rivera Jaramillo***, quienes me dieron las bases, y me inspiraron a entrar a una ingeniería, gracias a ellos logré llegar tan lejos.

Quiero agradecer a mi mejor amigo ***Antonio García Reyes***, quien me apoyo desde el bachillerato, siendo un amigo fiel, quien siempre me dio consejos, me escucho, y me enseño a relajarme en las circunstancias mas extremas, enseñándome que no todo en la vida es trabajo, si no tambien descansar y divertirse.

Nunca fui alguien de muchos amigos, pero agradezco mucho a aquellos que se quedaron ahí para apoyarme en los últimos años de mi licenciatura: ***Edna Daniela Escalona Villafuerte***, ***Ivonne Guillen Tinoco***, ***Jessica Paola Alcántar Mundo*** y ***Héctor Arturo Curiel Alvarado***; quienes me apoyaron en mis últimos años de carrera, y me impulsaron a terminarla dando lo mejor de mi en el camino, además de recibir su constante apoyo, incluso cuando era egoísta, y un mal amigo a veces, siempre me escucharon y me impulsaron, ***MUCHAS GRACIAS POR SU AMISTAD.***

CONTENIDO

Tabla de Figuras

CAPÍTULO 1: DEFINIENDO EL PROBLEMA

1.1 ¿Qué es el HMI y cuál es su importancia en el desarrollo de productos contemporáneos?

La *Human-Machine Interface* (HMI) o en español Interfaz Humano-Máquina, es la interacción que existe entre el ser humano con las máquinas, que permite manejar sistemas, así como el intercambio de información con estas. Unos ejemplos de estos elementos son los botones, pantallas táctiles, bocinas, etc. [7].

Hoy en día se pueden encontrar muchos ejemplos de HMI implementados en productos como celulares, controles remotos, y asistentes virtuales.

Figura 1: Asistentes virtuales existentes.

El enfoque más usual de interacciones HMI es mediante el tacto, por medio de botones o pantallas táctiles, no obstante, en años recientes se ha expandido a otras formas de interacción, como es el caso de la voz, un ejemplo de esto son los asistentes virtuales.

Los asistentes virtuales son utilizados también para crear casas inteligentes, en donde, con ayuda de dispositivos interconectados podemos controlar electrodomésticos, persianas, iluminación, etc. [20].

Por otra parte, la industria del entretenimiento usa otros enfoques de interacciones HMI, creando dispositivos hápticos para videojuegos, y dispositivos que permiten capturar nuestro entorno para la creación de películas con la técnica de captura de movimiento [45].

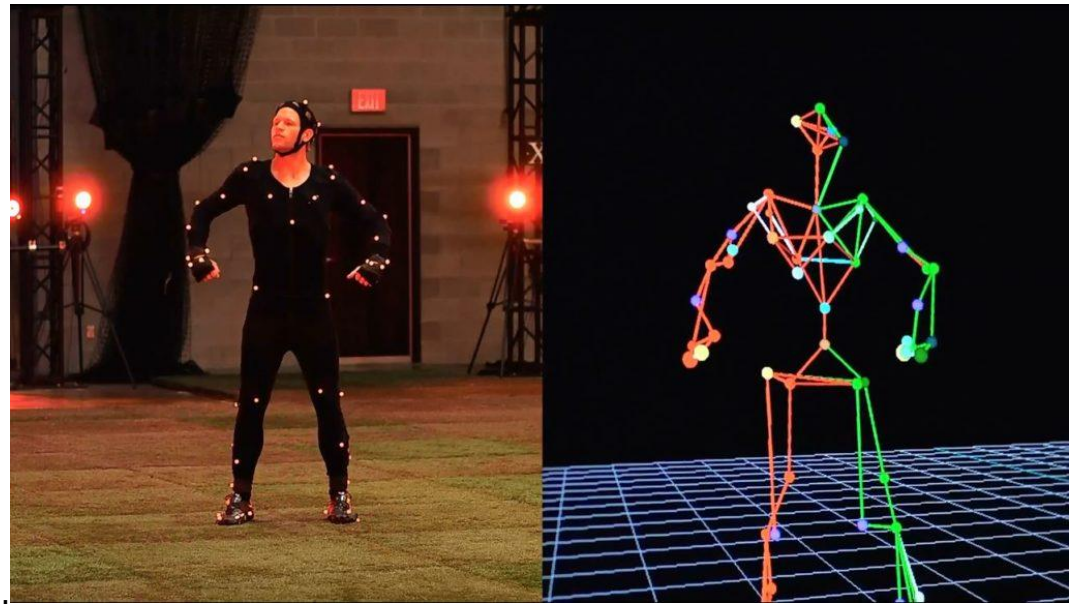


Figura 2: Ejemplo de la técnica de captura de movimiento.

Empresas como MANUS™ han desarrollado herramientas capaces de capturar los movimientos del cuerpo humano, para después ser presentados en un motor gráfico como Unity® o Unreal®, con la finalidad de mejorar la experiencia del usuario.

Un ejemplo de los productos de MANUS™ es su producto MetaGloves®, que se encarga de capturar el movimiento exclusivamente de la mano, con un costo de € 4 999 en marzo de 2023 [31].

Figura 3: Variantes de MetaGloves® MANUS™ [31].

La industria del entretenimiento no se enfoca en la interacción con las máquinas, sino en la forma en cómo estas ayudan al desarrollo de productos, dejando de lado el manejo de sistemas, pero no el intercambio de información.

El enfoque visual del HMI también se aplica a procesos de control utilizando visión artificial, permitiéndonos controlar otros dispositivos con el movimiento de cualquier parte del cuerpo, sin la necesidad de cargar ningún objeto adicional en el proceso. [14].

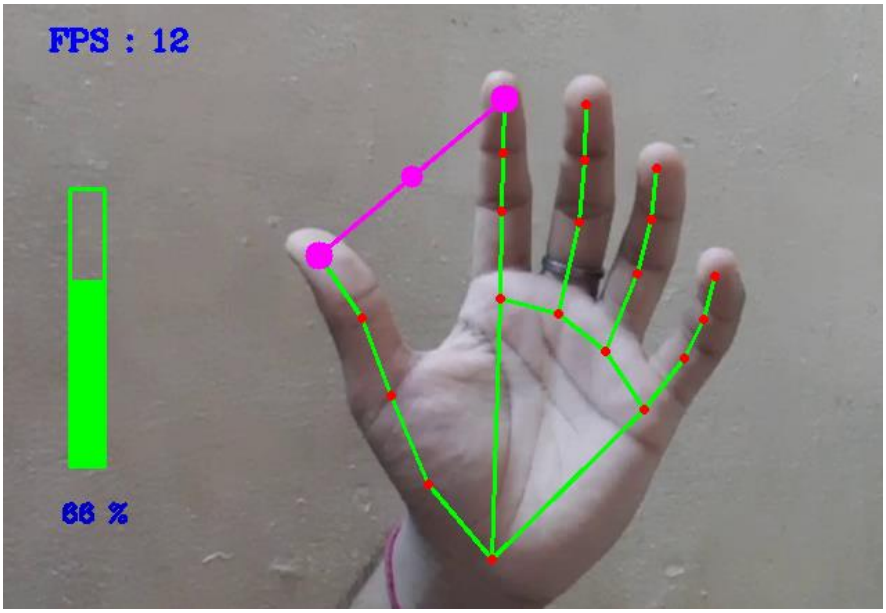


Figura 4: Ejemplo de visión artificial con Python [14].

Todas las interacciones tienen algo en común, y es que necesitan de la presencia cercana de una persona, ya sea con la voz, una cámara de video, o el tacto de un dedo. Por lo que nace la pregunta, **¿Puede existir otros enfoques de HMI que no necesiten de nuestra presencia cercana?**

1.2 Análisis PESTEL de la problemática

Es posible encontrar nuevas formas de HMI basándonos en propuestas existentes, donde el usuario pueda controlar dispositivos interconectados con una mano.

Para tener un mejor contexto de la problemática, es necesario realizar un análisis de los factores externos que pueden afectar al proyecto, para eso se optó por realizar un análisis PESTEL, el cual nos permite revisar los factores Políticos, Económicos, Sociales, Tecnológicos, Ecológicos y Legales de los dispositivos HMI en general:

Normativas

Algunas de las normativas en el diseño del HMI incluyen la ISA-101, ISO-13407 y IEC-61499 (pantallas, terminales de operador y sistemas SCADA), además de las normativas generales para los dispositivos electrónicos [18].

La ergonomía es uno de los factores más importantes en el diseño de dispositivos HMI, en donde se busca garantizar la seguridad y eficiencia del usuario, para esto, los dispositivos necesitan ser intuitivo y fáciles de utilizar para minimizar el estrés y la fatiga del usuario [19].

Económico

La pandemia mundial por COVID-19 provoco la ruptura de cadenas productivas y de manufactura, entre ellas la de semiconductores, lo que afectó a la fabricación de artículos como tabletas, automóviles, teléfonos, computadoras, etc.

En 2021, el 64% de las compañías en el sector de las Tecnologías de la Información y la Comunicación (TIC) aseguraron que hubo un problema de desabasto de productos, afectando la demanda del consumidor, por lo tanto, afectando su economía [13].

Social

Si llega a existir un nuevo enfoque del HMI, su reacción en los usuarios puede tener un peso importante en como el diseño se puede desarrollar a largo plazo.

El incremento de casas inteligentes ha tenido un peso muy grande después de la pandemia, con un total de total de 2.79 millones de hogares lo

implementan en México, de un total de 35 millones de viviendas en 2023, y se espera un incremento de éstas del 19% para 2024 [11].

Tecnológico

La obsolescencia tecnológica es un factor importante que puede afectar al diseño de nuevos dispositivos, cambiando su hardware, para implementar o quitar funciones. Es recomendable estar actualizado en las nuevas tecnologías para evitar este fenómeno, además de aprovechar las ventajas que conlleva, como funciones mejoradas [34].

La mayoría de las personas pueden ver a una velocidad de fotogramas entre 30 y 60 FPS, pero es más común en videojuegos donde se necesite precisión usar 60 FPS para no tener perdida de información y mejor percepción del entornó [21]

Ecológico

La impresión de placas PCB ha demostrado generar subproductos tóxicos que pueden dañar a diferentes organismos, provocando alteraciones neurodegenerativas [32].

Político

Existen en el mercado existen productos que cuentan con patentes, por lo que se tiene que contemplar para el diseño de un nuevo dispositivo si están patentadas ideas similares.

Figura 5: Análisis PESTEL del proyecto.

1.3 Objetivos y alcances del trabajo

Los objetos del proyecto se dividen en dos: Objetivo general y los objetivos particulares:

Objetivo General:

Minimizar los tiempos de procesamiento de un microcontrolador ESP32, para generar una experiencia de usuario lo más cercano al concepto de "tiempo real", en el contexto de adquisición de señales para gestos de mano.

Objetivos particulares:

1. Desarrollar un modelo cinemático para la mano que reduzca el número de variables de salida en aras de optimizar los tiempos de procesamiento en el ESP32.
2. Implementar dicho modelo cinemático en el diseño de configuración de un dispositivo de adquisición de datos.
3. Diseñar hasta la etapa de detalle un dispositivo para la adquisición de datos para gestos de la mano, en el contexto del HMI (“Human-Machine Interface”).

Los alcances de este proyecto incluyen la creación de un modelo funcional que permita validar el modelo cinemático propuesto. Este modelo funcional servirá como una primera iteración para el diseño de un producto en forma.

Como primera etapa, se buscará determinar la viabilidad de integrar la instrumentación necesaria en un guante. Además, se demostrarán las capacidades de las tecnologías disponibles en la actualidad para crear un dispositivo capaz de actualizarse de forma remota, comunicarse con otros dispositivos de manera inalámbrica y procesar la información de los sensores en un tiempo razonable.

CAPÍTULO 2: DISEÑO CONCEPTUAL

2.1 Análisis del espacio de soluciones y diagramas de polaridad

Los productos electrónicos existentes implementan formas de HMI, pero muy pocos se centran en mejorar esta interacción, los más comunes son objetos enfocados en el entretenimiento, utilizando botones, joysticks, y pantallas táctiles.

Las funciones del sistema son las siguientes:

- ***Comunicar*** (a las personas)
- ***Interactuar*** (con dispositivos)
- ***Detectar*** (gestos/acciones)
- ***Muestrear*** (los datos)
- ***Optimizar*** (el proceso)

Para completar el análisis del espacio de soluciones es necesario desarrollar diagramas de polaridad basados en una lista de especificaciones del sistema que cumplan con las funciones del sistema, en donde podemos buscar posibles áreas de oportunidad:

- Precisos
- Exactos
- Sentido del tacto
- Sentido del oído
- Necesitan de nuestra presencia
- Cuentan con dependencias externas (un objeto adicional al sistema)
- Pesados
- Ligeros

Los diagramas de polaridad son con base a una investigación de productos existentes en el mercado, en donde podemos encontrar tecnologías de todo tipo, como asistentes virtuales, controles remotos, juguetes, dispositivos industriales y proyectos caseros.

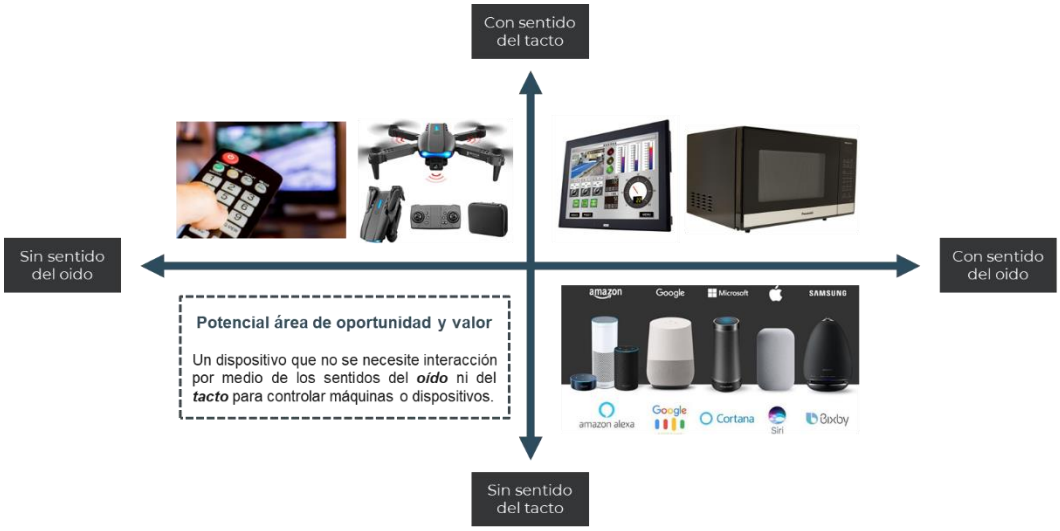


Figura 6: Diagrama de polaridad 1: Tacto y oído.

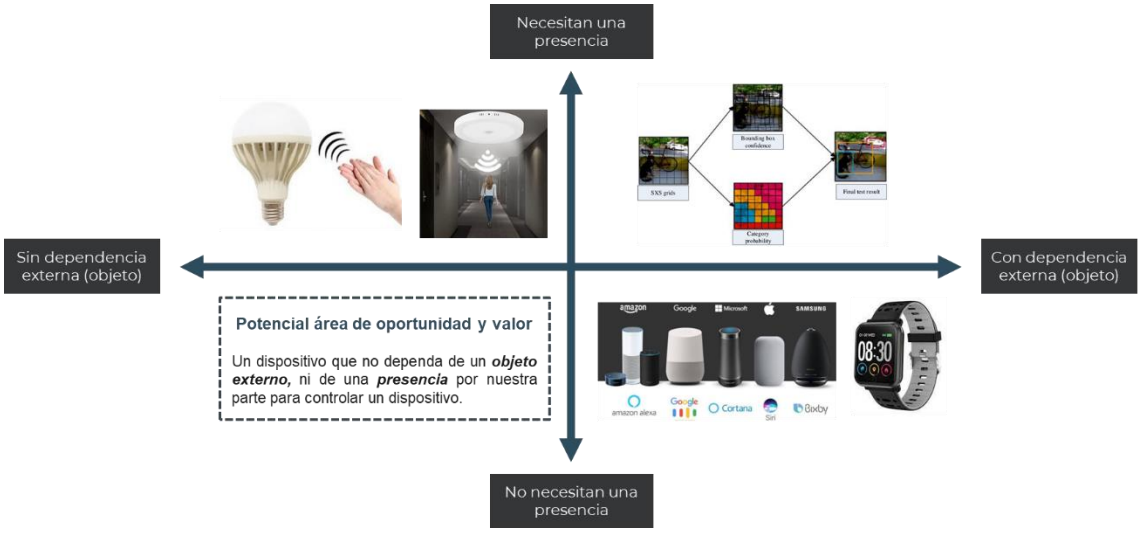


Figura 7: Diagrama de polaridad 2: Presencia y dependencias.

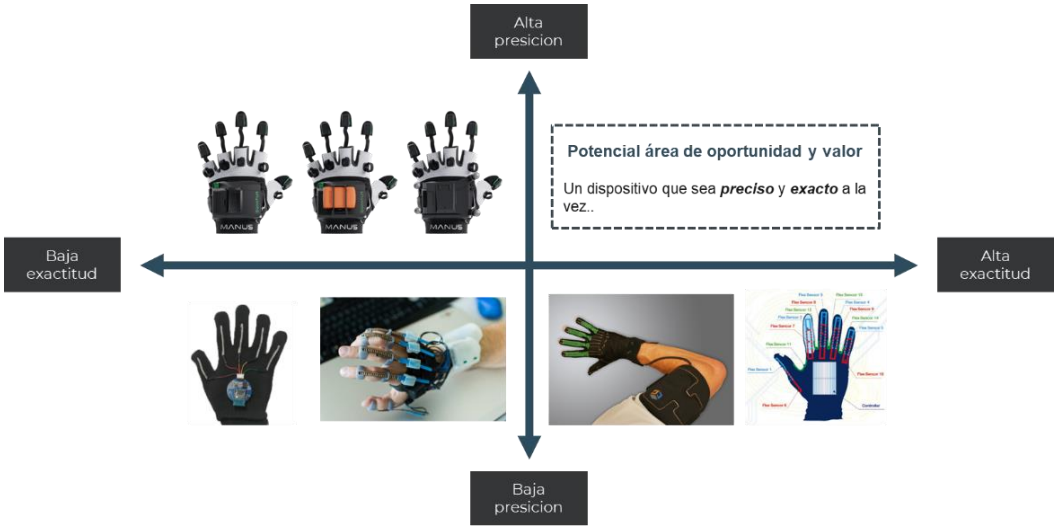


Figura 8:Diagrama de polaridad 3: Exactitud y precisión.

Con base en los diagramas de polaridad se puede encontrar atributos para la propuesta de valor para el proyecto:

- Un proyecto que no necesite de los sentidos del tacto ni del oído tanto del usuario como del dispositivo.
- Un proyecto que no necesite de un objeto externo ajeno al sistema, de tal forma que no tengamos contacto de ningún tipo con otro objeto, además de que no necesita de nuestra presencia.
- Un dispositivo que se exacto y preciso a la vez.

2.2 Requerimientos y especificaciones objetivo

A partir de las funciones del sistema podemos desarrollar los requerimientos y especificaciones objetivo de este proyecto:

Requerimientos:

1. El sistema comunicará a las personas con los dispositivos.
2. El sistema interactuará con dispositivos variados.
3. El sistema detectará gestos y acciones mediante ángulos.
4. El sistema debe tener una comunicación inalámbrica.
5. El sistema debe tener una alta rapidez de procesamiento.
6. El sistema debe tener frecuencia de muestreo altas.

En función a los requerimientos y a una investigación previa de sensores y dispositivos existentes se pueden plantear las especificaciones del sistema:

- Para el requerimiento 3:
  - Resolución angular medir menor a 1 [°] (referencia: unidad en grados).
  - Rapidez del sensor mayor a 250 [°/s] (valor común en la industria).
- Para el requerimiento 4:
  - Distancia menor a 5 [m] (valor arbitrario).
  - Tasa de transferencia remota igual a 11520 [baudios] (valor común en dispositivos).
- Para el requerimiento 5:
  - Tasa de procesamiento igual o mayor a 60 [Hz] (valor arbitrario no estrictamente necesario, se decidió tomar un valor equivalente 60 FPS, referencia: videojuegos).
  - La cantidad de datos de salida (modelo cinemático).
- Para el requerimiento 6:
  - La tasa de muestreo debe ser mayor a 60 Hz (mayor al tiempo de procesamiento).

2.3 Desarrollo de conceptos de solución

En la actualidad, existen diseños preestablecidos de guantes que utilizan sensores de flexión resistivos. La exactitud de estos guantes puede variar considerablemente y depende de factores como la cantidad de sensores utilizados y su ubicación en la mano.

Estos diseños basados en sensores resistivos son particularmente comunes en proyectos más pequeños debido a la facilidad de adquisición y uso de sus componentes. La salida analógica de los sensores puede ser interpretada de diversas maneras, lo que resulta en una medición de la flexión y extensión del dedo completo en dispositivos con pocos sensores, sin la necesidad de crear algoritmos complejos o utilizar fórmulas extensas.

Una ventaja de este diseño es que los guantes son más ligeros y fáciles de confeccionar. Simplemente se cosen estos sensores en el guante, ubicándolos en posiciones que coinciden con las articulaciones interfalángicas y metacarpofalángicas.

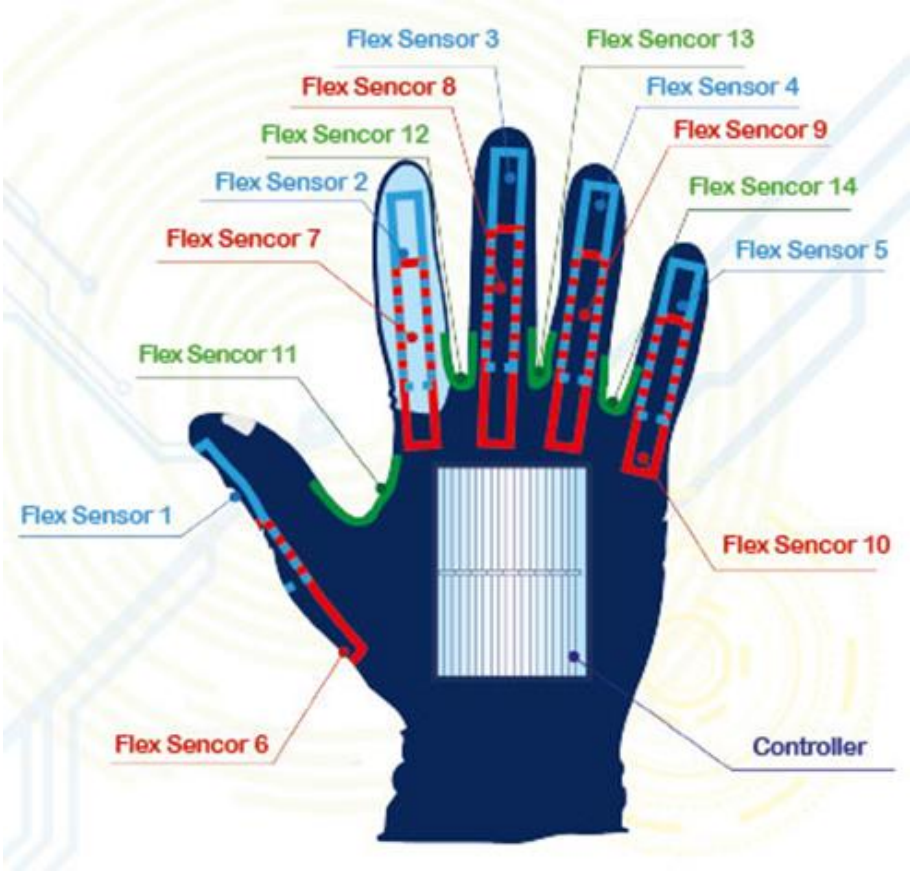


Figura 9: Diseño de guante que traduce lenguaje de signos con base en sensores resistivos tipo Flex [35].

Otras variantes de guantes actuales utilizan sensores IMU para medir la orientación de ciertas partes de la mano, como la palma y algunas falanges. Estos diseños son más complejos de entender a grandes rasgos, ya que además del hardware, se complementan con software que realiza cálculos para recrear el movimiento de los dedos. Por lo general, este tipo de diseños tiende a ser más costoso.

Un ejemplo destacado de esto son los guantes de MANUS™, que implementan IMU en sus guantes, aunque también utilizan un número limitado de sensores de flexión para medir movimientos específicos. Estos guantes han tenido un gran impacto en el mercado y han obtenido un amplio soporte en diferentes motores gráficos. Además, han sido el resultado de varios años de desarrollo para alcanzar su producto final.

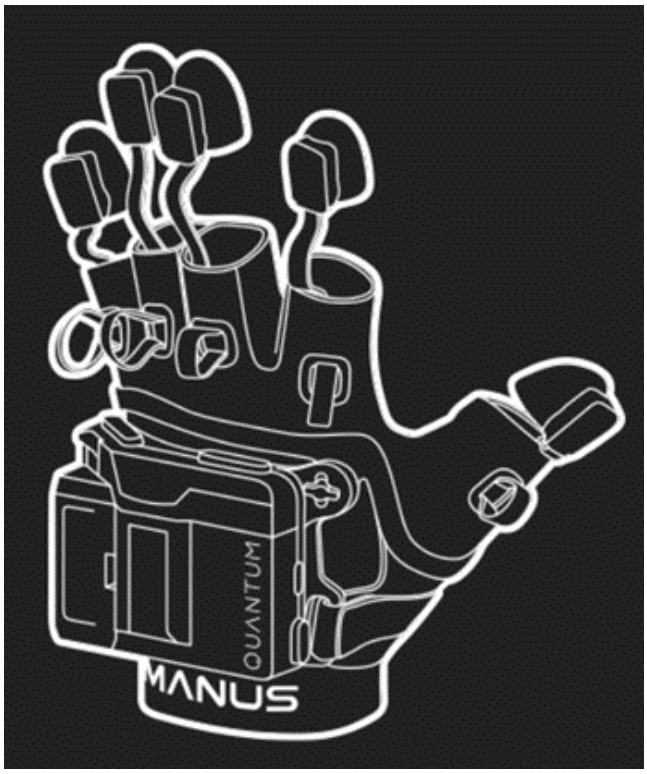


Figura 10: Ejemplo de guante MetaGloves® implementando IMU [31].

En general, los diseños actuales se basan en tecnologías como las IMU y sensores de flexión, y han estado utilizando estas técnicas durante años, logrando resultados significativos en la modelación de la mano. Sin embargo, también presentan ciertos errores menores que pueden afectar la precisión y hacer que estos dispositivos se alejen ligeramente de la realidad. Dado que los diseños basados en las IMU y los sensores Flex son bastante similares, no se busca necesariamente innovar en este tipo de tecnologías.

2.4 Análisis y selección del concepto final

Basándose en el desarrollo de conceptos de solución, se puede concluir que las tecnologías actuales son adecuadas para el diseño de guantes, pero presentan ciertas desventajas. En el caso de los sensores de flexión resistivos, estos pueden deformarse permanentemente, lo que cambia sus valores y fórmulas predefinidos, además de que son propensos a la rotura con el tiempo, lo que hace que sean menos viables debido a estas desventajas evidentes.

Por otro lado, los guantes basados en IMU son más precisos, aunque su manipulación puede ser complicada. Ofrecen mejores resultados en términos de precisión, además es importante tener en cuenta que estos sensores tienden a tener una pequeña deriva después de un uso prolongado, lo cual es un error aceptable. Por ejemplo, en ángulos de Euler, el MPU-6050 demostró tener una deriva de 1 grado cada 5 minutos con un sensor económico.

Algunos productos actuales, como los guantes MANUS™, incluso incorporan una sola IMU en las falanges distales. Esto sugiere que se utilizan paquetes de software para modelar la mano, como la paquetería **Inverse Kinematics** en Unity. Sin embargo, esto puede llevar a una pérdida de exactitud, como se observa en la Figura 11: Captura del video de los MetaGloves® de MANUS™ [31]. donde la posición de los dedos no es completamente igual.

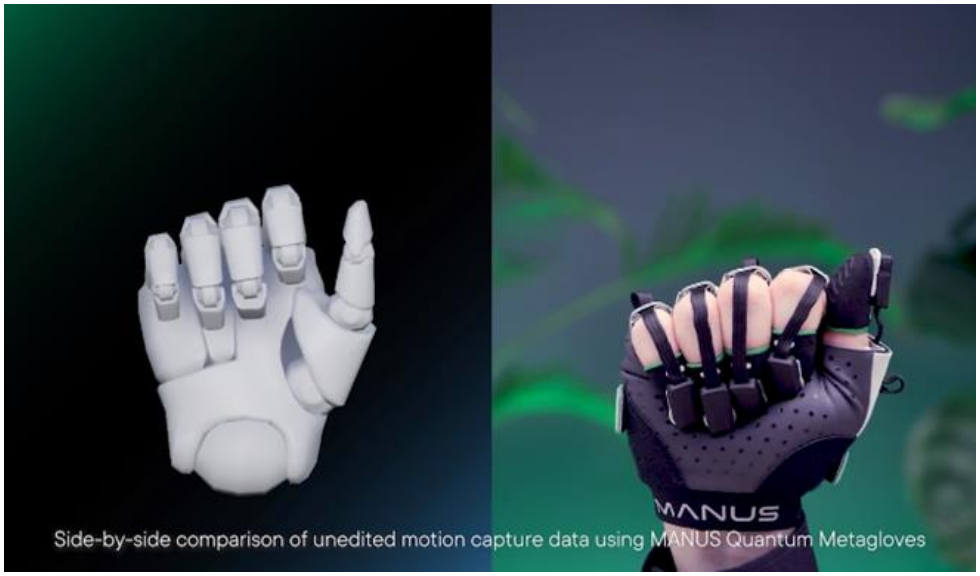


Figura 11: Captura del video de los MetaGloves® de MANUS™ [31].

Para evitar este tipo de error, se propone un diseño alternativo en el que se emplea un sensor tipo IMU en cada falange, similar al diseño con sensores de flexión resistivos, pero sustituyendo el tipo de sensor. Esto permitiría lograr una mayor precisión en cada falange, ya que los sensores IMU ofrecen una mejor precisión en comparación con los sensores de flexión.

Este diseño se basa en la idea de que cada articulación interfalángica y metacarpofalángica tiene movimientos pasivos. Al tener más sensores, se obtiene una mayor combinación de movimientos posibles, lo que aumenta la capacidad de modelar de manera más precisa la mano y sus articulaciones.

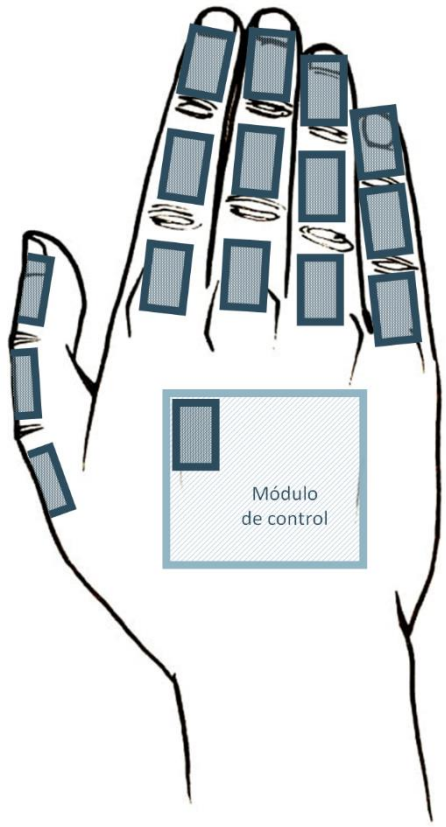


Figura 12: Concepto final de guante, en donde los recuadros azules representan los sensores IMU.

En el módulo de control se encontrará el microcontrolador que se encargará de procesar la información de los sensores, así como circuitos auxiliares, como un módulo bluetooth para la comunicación inalámbrica y multiplexores, dada la gran cantidad sensores iguales.

CAPÍTULO 3: ESTADO DEL ARTE

3.1 Aspectos anatomofisiológicos de la mano

La mano es una estructura altamente compleja que posee la capacidad de realizar una amplia variedad de movimientos y adoptar numerosas posiciones que son fundamentales en nuestras actividades diarias, como escribir, manipular objetos, cocinar, entre otras.

Para comprender su funcionamiento a un nivel general, es esencial conocer las partes que la componen. Sin embargo, modelar la mano en su totalidad se presenta como un desafío considerable, esto se debe a que el par articular resultante puede ser generado por múltiples contribuciones relativas de diferentes músculos individuales, lo que se conoce como redundancia muscular [47].

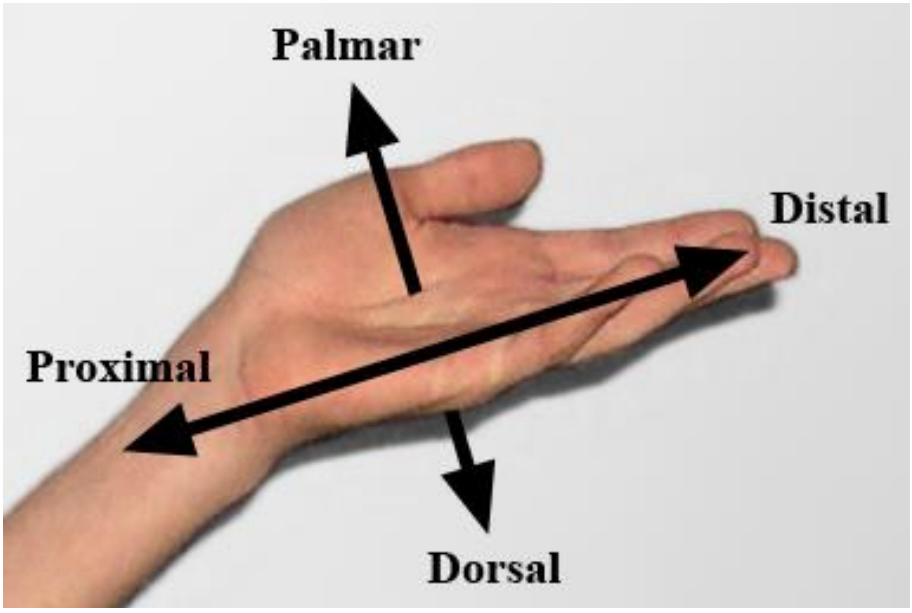


Figura 13: Referencia anatómica de la mano [48].

HUESOS

La mano en conjunto con la muñeca tiene en total de 27 huesos (por cada mano), y a su vez esta se subdivide en 3 grupos de huesos:

- ***Falanges***: Contamos con un total de 14 huesos que están en los dedos de la mano. Estos a su vez se dividen en 3 falanges (distal, media y proximal)
- ***Metacarpianos***: Contamos con 5 huesos que conforman la parte media de la mano.
- ***Carpianos***: Contamos con 8 huesos que forman parte de la muñeca.

Figura 14: Anatomía de la mano [41].

MÚSCULOS

Los músculos nos ayudan a efectuar ciertos movimientos del cuerpo, los principales de la mano son:

- ***Abducción***: Movimiento en el que dos miembros se alejan.
- ***Aducción***: Movimiento en el que dos miembros se acercan.
- ***Flexión***: Disminución del ángulo que existe entre dos huesos.
- ***Extensión***: Aumento del ángulo que existe entre dos huesos.
- ***Oposición***: Tocar con el pulgar los demás dedos (Exclusivo del pulgar).
- ***Reposición***: Separar el pulgar de los demás dedos (Exclusivo del pulgar).
- ***Supinación***: Rotar el antebrazo para que la palma de la mano quede mirando hacia arriba. (Exclusivo del antebrazo).

- ***Pronación***: Rotar el antebrazo para que la palma de la mano quede mirando hacia abajo. (Exclusivo del antebrazo).

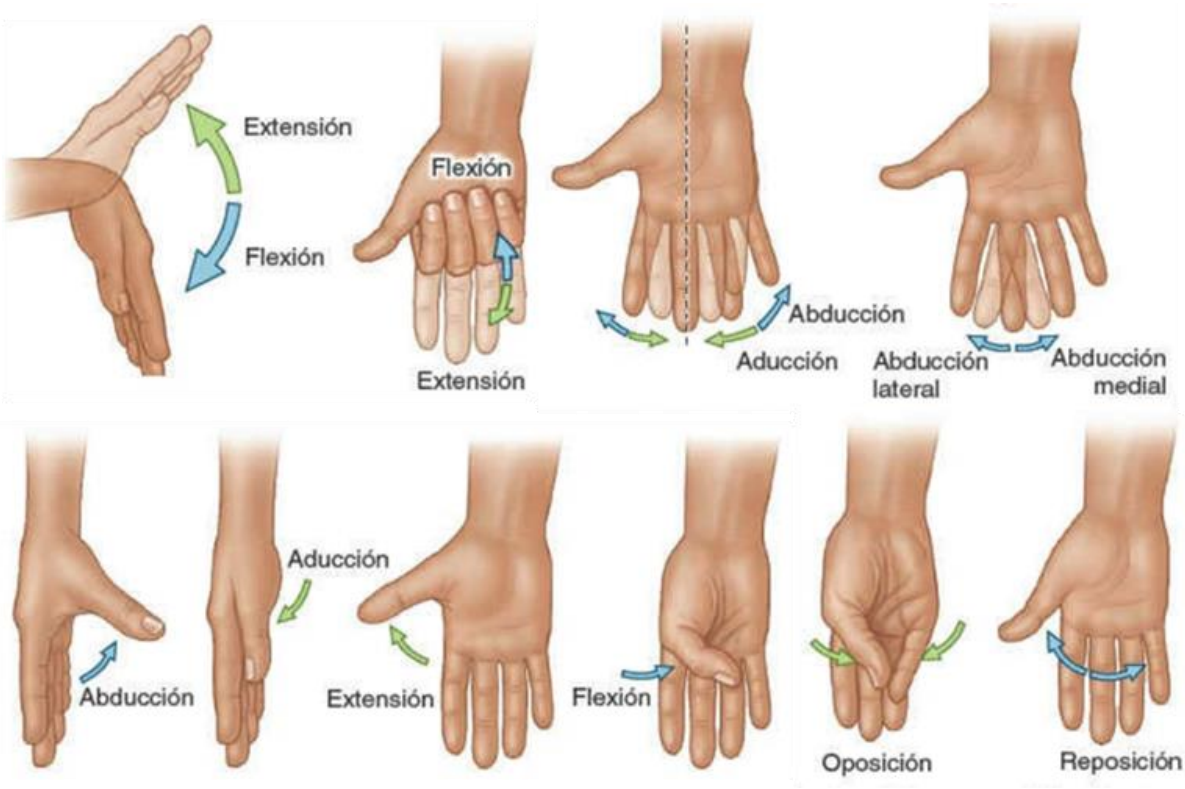


Figura 15: Movimientos de la mano [1].

Los músculos de la mano están divididos en 2 grupos:

- ***Extrínsecos***: Estos músculos están en los antebrazos y proporcionan fuerza y agarre.
- ***Intrínsecos***: Estos músculos están dentro de la mano y se encargan del movimiento fino de los dedos, estos a su vez se dividen en 4 subgrupos.
    - ***Músculos tenares***: Son los músculos encargados de la abducción, aducción, flexión, extensión, oposición y reposición del pulgar.
    - ***Músculos hipotenares***: Son los músculos encargados de la abducción, aducción, flexión, extensión, oposición y reposición del dedo meñique.
    - ***Lumbricales***: Son los músculos encargados de la flexión y extensión de los dedos 2do-5to.
    - ***Interóseos***: Son los músculos encargados de la abducción y aducción de los dedos 2do-5to.

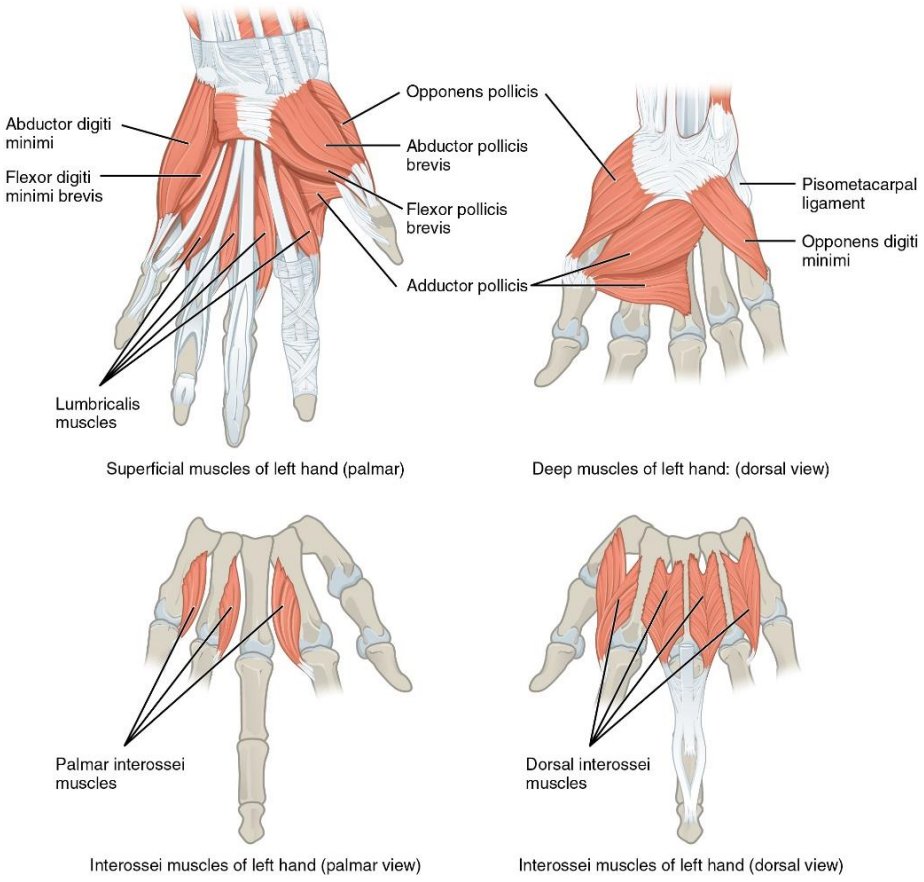


Figura 16: Músculos intrínsecos de la mano [44].

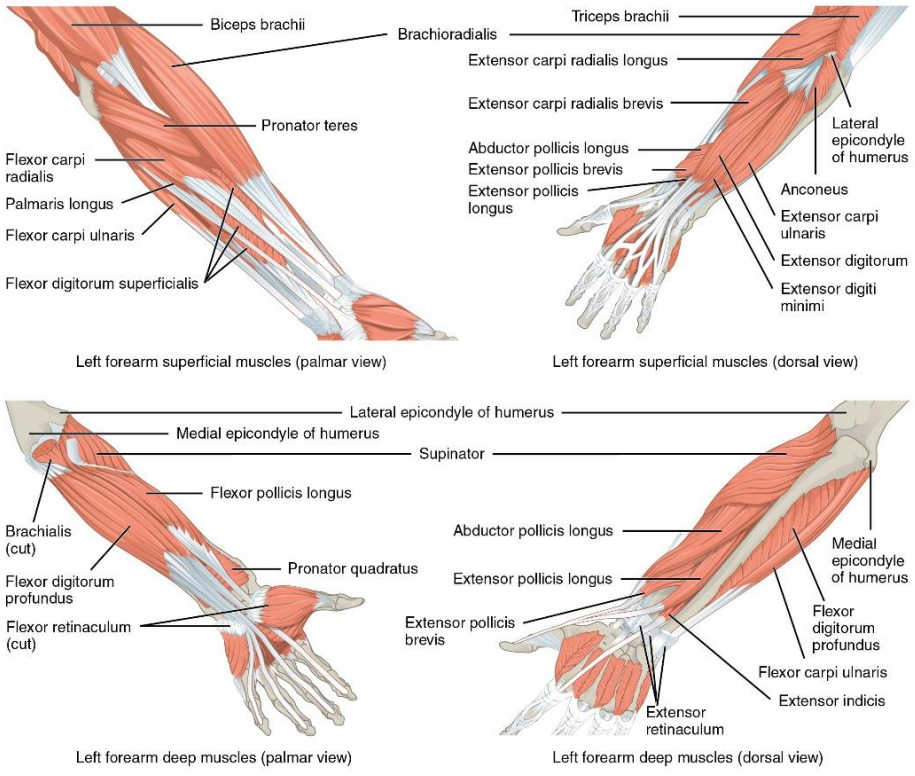


Figura 17: Músculos extrínsecos de la mano [44].

ARQUITECTURA DE LA MANO

La mano en conjunto con los dedos tiene muchas posiciones para diferentes aplicaciones, como lo es agarrar un objeto voluminoso.

Ciertas posiciones de la mano nos ayudan para ver como son la forma de los dedos al agarrar objetos y cuando buscamos hacer gestos. Es importante ver los puntos de referencia que tiene la mano, para ver más a detalle sus funciones, destacan 4 posiciones:

- Dedos separados voluntariamente (a): Vemos que las líneas de los dedos se intersecan cerca de la muñeca.
- Puño (b): Vemos que las líneas de los dedos cambian su intersección, esta intersección se encuentra en la muñeca, del lado del pulgar.
- Posición natural (c): Vemos que las líneas de los dedos 3-5 son paralelas, mientras que el índice tiene una inclinación natural, al igual que el pulgar.
- Dedos aproximados voluntariamente (d): Vemos que las líneas de los dedos tienen una cierta inclinación, estas convergen casi al infinito debido a la poca inclinación que tienen entre sí.

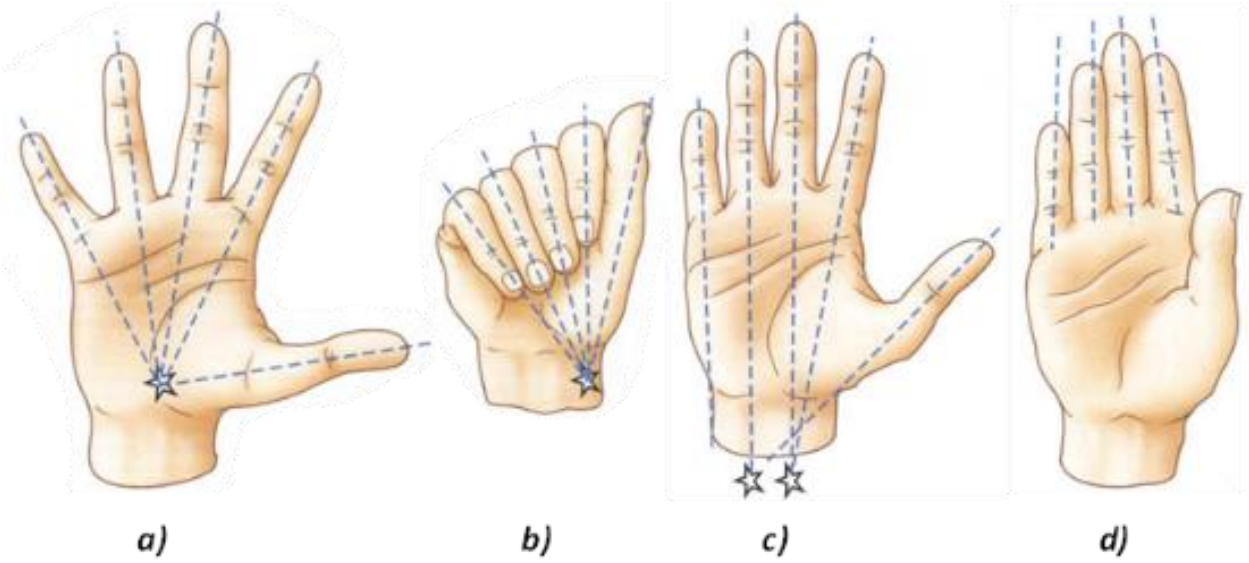


Figura 18: Posiciones de la mano [22].

Por último, todos los dedos cuentan con una flexión oblicua, permitiendo que el dedo meñique pueda tocar el pulgar sin mucha dificultad, de ahí la oposición y reposición del pulgar y el meñique.

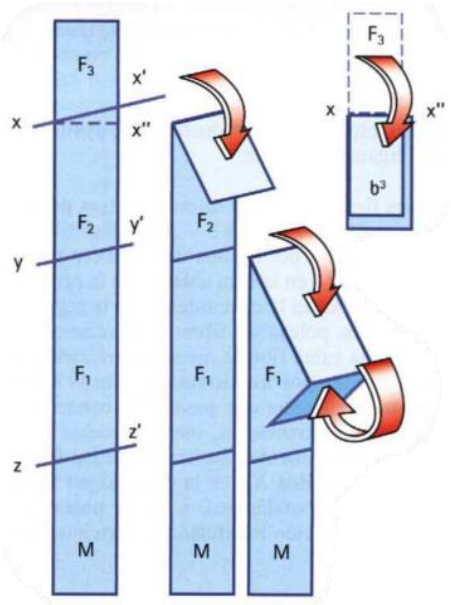


Figura 19: Ejemplificación de cómo funcionan las falanges [22].

ARTICULACIONES INTERFALÁNGICAS Y METACARPOFÁLANGICAS

Las articulaciones de los dedos desempeñan un papel fundamental al permitirnos realizar movimientos de las falanges, a menudo comparados con el funcionamiento de poleas. Estas articulaciones se han documentado y se pueden clasificar en dos tipos distintos:

- **Activas**: Es la amplitud máxima realizada conscientemente por una persona
- **Pasiva**: Es la amplitud máxima que se realiza con ayuda de una fuerza externa

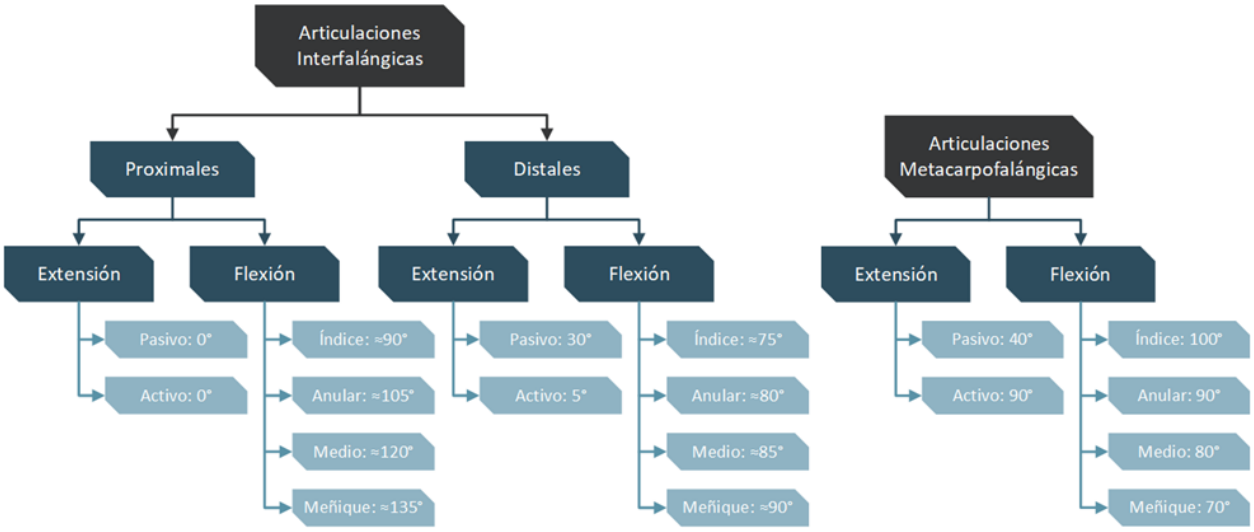


Figura 20: Diagrama de amplitudes de las articulaciones de la mano.

3.2 Cinemática de un cuerpo rígido.

Esta sección explicará únicamente la posición y orientación de los cuerpos rígidos, así como las dos técnicas para el análisis de una cadena cinemática: Cinemática Inversa y cinemática Directa.

POSICIÓN Y ORIENTACIÓN DE LOS CUERPOS RÍGIDOS

Un ***cuerpo rígido*** puede considerarse como una combinación de un gran número de partículas donde todas estas permanecen a una distancia fija entre sí, tanto antes como después de la aplicación de una carga [16].

Podemos describir un cuerpo rígido en el espacio $\mathbb{R}^3$ por medio de su posición y orientación, en donde la posición se representa con un vector y su orientación con una matriz de rotación.

Las matrices de rotación están compuestas por 3 vectores columna, esta matriz nos permite expresar la orientación de un marco coordenado B con respecto a un marco A [39]:

$$c_x = \begin{pmatrix} \hat{\imath}_B \cdot \hat{\imath}_A \\ \hat{\imath}_B \cdot \hat{\jmath}_A \\ \hat{\imath}_B \cdot \hat{k}_A \end{pmatrix} \quad c_y = \begin{pmatrix} \hat{\jmath}_B \cdot \hat{\imath}_A \\ \hat{\jmath}_B \cdot \hat{\jmath}_A \\ \hat{\jmath}_B \cdot \hat{k}_A \end{pmatrix} \quad c_z = \begin{pmatrix} \hat{k}_B \cdot \hat{\imath}_A \\ \hat{k}_B \cdot \hat{\jmath}_A \\ \hat{k}_B \cdot \hat{k}_A \end{pmatrix}$$

$$\overset{A}{B}\boldsymbol{R} = \begin{pmatrix} c_x & c_y & c_z \end{pmatrix}$$

$$\overset{A}{B}\boldsymbol{R} = \begin{pmatrix} \hat{\imath}_B \cdot \hat{\imath}_A & \hat{\jmath}_B \cdot \hat{\imath}_A & \hat{k}_B \cdot \hat{\imath}_A \\ \hat{\imath}_B \cdot \hat{\jmath}_A & \hat{\jmath}_B \cdot \hat{\jmath}_A & \hat{k}_B \cdot \hat{\jmath}_A \\ \hat{\imath}_B \cdot \hat{k}_A & \hat{\jmath}_B \cdot \hat{k}_A & \hat{k}_B \cdot \hat{k}_A \end{pmatrix} \tag{3.1}$$

Una matriz de rotación tiene las siguientes propiedades [39]:

- Son una matriz de 3x3.
- Sus vectores columna son ortogonales.

$$c_x \cdot c_y = c_x \cdot c_z = c_y \cdot c_z = 0$$

- Sus vectores columna tienen magnitud unitaria.

$$\|c_x\| = \|c_y\| = \|c_z\| = 1$$

- Su determinando es igual a 1.

$$\det(\boldsymbol{R}) = +1$$

- Su matriz inversa es igual a la transpuesta.

$$\mathbf{R}^{-1} = \boldsymbol{R}^T$$

Existen 3 rotaciones principales: rotación sobre el eje **X**, rotación sobre el eje **Y** y rotación sobre el eje **Z**.

$$\boldsymbol{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} \quad \textbf{(3.2)}$$

$$\boldsymbol{R}_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \quad \textbf{(3.3)}$$

$$\boldsymbol{R}_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \textbf{(3.4)}$$

Las matrices de rotación se pueden multiplicar entre sí para generar nuevas rotaciones. Es muy importante verificar el orden de las matrices, ya que pueden generar rotaciones erróneas.

Para transformar las coordenadas de un vector, basta con multiplicar la matriz por el vector en forma de matriz 3x1 [39].

$$^{A}\vec{v} = {}^{A}_{B}\boldsymbol{R} \cdot {}^{B}\vec{v}$$

CADENA CINEMÁTICA

Una ***cadena cinemática*** es un conjunto de cuerpos rígidos conectados por juntas para proporcionar un movimiento restringido que es el modelo matemático de un sistema mecánico [4].

En una cadena cinemática podemos agrupar las variables en dos grupos:

- **Coordenadas articuladas:** Los ángulos de las articulaciones de los eslabones. Valores relativos.
- **Pose del efecto final:** La posición y orientación de la cadena cinemática. Valores absolutos.

Existen dos métodos para calcular todas las variables en función de las variables conocidas:

Cinemática Directa

Es determinar la pose del efecto final en función de las coordenadas articuladas [4].

Cinemática Inversa

Es determinar las coordenadas articuladas en función de la pose del efecto final [4].

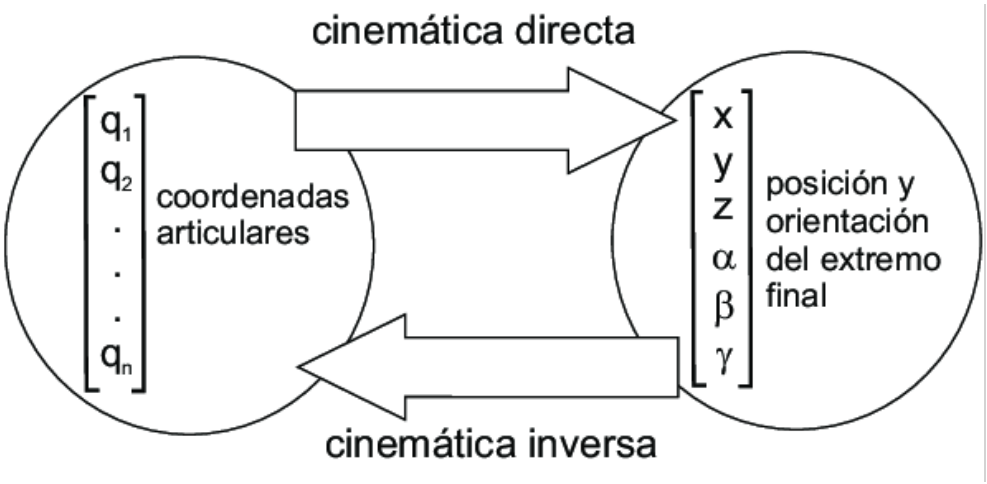


Figura 21: Diagrama de relación entre cinemática directa e inversa [4].

DESCRIPCIÓN CINEMÁTICA A TRAVÉS DE CUATERNIONES.

El termino cuaternión etimológicamente proviene del latín ***quaterni*** (por cuatro); cuyo significado es "conjunto de cuatro elementos. Los cuaterniones fueron creados en 1843 por William Rowan Hamilton, el cual buscaba realizar rotaciones en el espacio $\mathbb{R}^3$ con números complejos. Viene dada por la expresión [40].

$$\boldsymbol{q} = a + b\hat{\boldsymbol{\imath}} + c\hat{\boldsymbol{\jmath}} + d\hat{\boldsymbol{k}} \tag{3.5}$$

Los cuaterniones cuentan con 3 unidades complejas representadas por ***i***, ***j*** y ***k*** tal que:

$$\hat{\boldsymbol{\imath}}^2 = \hat{\boldsymbol{\jmath}}^2 = \hat{\boldsymbol{k}}^2 = \hat{\boldsymbol{\imath}}\hat{\boldsymbol{\jmath}}\hat{\boldsymbol{k}} = -1$$

Éstos al ser una extensión de los números complejos, tienen propiedades algebraicas similares, la única diferencia es la multiplicación, que no es conmutativa.

$$q_1 = a_1 + b_1\hat{\boldsymbol{\imath}} + c_1\hat{\boldsymbol{\jmath}} + d_1\hat{\boldsymbol{k}}$$

$$q_2 = a_2 + b_2\hat{\boldsymbol{\imath}} + c_2\hat{\boldsymbol{\jmath}} + d_2\hat{\boldsymbol{k}}$$

$$\begin{aligned} q_1 \cdot q_2 = {} & (a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2) + (a_2b_1 + b_2a_1 + c_2d_1 - d_2c_1)\hat{\boldsymbol{\imath}} \\ & + (a_2c_1 - b_2d_1 + c_2a_1 + d_2b_1)\hat{\boldsymbol{\jmath}} + (a_2d_1 + b_2c_1 - c_2b_1 + d_2a_1)\hat{\boldsymbol{k}} \end{aligned} \tag{3.6}$$

Multiplicando un cuaternión arbitrario por un cuaternión únicamente con parte real unitaria, se obtiene el mismo cuaternión:

$$q_1 \cdot \left(1 + 0\hat{\boldsymbol{\imath}} + 0\hat{\boldsymbol{\jmath}} + 0\hat{\boldsymbol{k}}\right) = q_1$$

Multiplicar un cuaternión por su conjugado da como resultado la siguiente expresión:

$$q_1 \cdot \bar{q}_1 = a^2 + b^2 + c^2 + d^2$$

En el caso de que los cuaterniones fueran unitarios, la propiedad anterior daría como resultado 1, lo que también se podría interpretar como un cuaternión con únicamente parte real.

En general, esta metodología suele ser más corta a comparación de las matrices de rotación, debido a la diferencia entre la cantidad de datos que se necesitan para formular una rotación, en donde una matriz necesita 9 datos para definir una rotación en $\mathbb{R}^3$, mientras que un cuaternión solo necesita 4.

Para representar una rotación con un cuaternión se necesita un vector $\vec{\boldsymbol{v}}$ en el espacio que representa nuestro eje de rotación, además de un ángulo de rotación, dicha expresión es la.

$$\vec{\boldsymbol{v}} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

$$\boldsymbol{q}(\theta, \vec{v}) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ v_1 \cdot \sin\left(\frac{\theta}{2}\right) \\ v_2 \cdot \sin\left(\frac{\theta}{2}\right) \\ v_3 \cdot \sin\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{3.7}$$

Para cambiar el sentido de una rotación se usa la matriz transpuesta o inversa, pero, en el caso de cuaterniones, se usa su vector conjugado.

Se puede crear una matriz de rotación en función de un cuaternión con la fórmula [40].

$$\boldsymbol{R}(\vec{q}) = \begin{pmatrix} 2q_0^2 + 2q_1^2 - 1 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & 2q_0^2 + 2q_2^2 - 1 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & 2q_0^2 + 2q_3^2 - 1 \end{pmatrix} \tag{3.8}$$

Para realizar rotaciones de vectores con cuaterniones se tiene que convertir el vector a un cuaternión igualando la parte real a 0 y sustituyendo la parte compleja del cuaternión por cada uno de sus valores en XYZ respectivamente [40]:

$$\boldsymbol{q}(\vec{v}) = \begin{pmatrix} 0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} \tag{3.9}$$

CAPÍTULO 4: EMBODIMENT Y MODELO CINEMÁTICO DE LA MANO

4.1 Modelos técnicos

De acuerdo con los requerimientos y especificaciones objetivo, se busca que el sistema tenga una frecuencia de iteración igual o superior a 60 [Hz].

Para alcanzar una frecuencia superior a 60 [Hz] (equivalente a un tiempo de iteración de 16.66 [ms]), es necesario considerar factores como la velocidad de transferencia, la cantidad y tipo de datos, la frecuencia de muestreo y el tiempo de procesamiento. Estas consideraciones varían en función del microcontrolador utilizado, el sensor específico y el algoritmo implementado.

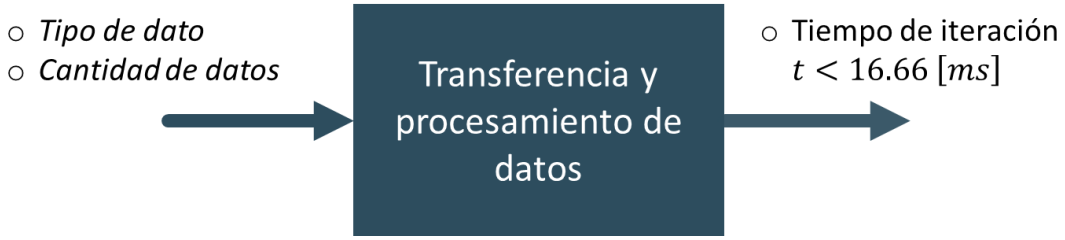


Figura 22: Modelo técnico 1 - Transferencia y procesamiento de datos.

En el procesamiento de datos, está implícito el uso de algoritmos para fusionar los datos del IMU, conocidos como ***filtros complementarios***, que nos proporcionan la orientación de un objeto en función de los cuaterniones. Actualmente existen bastantes filtros, basados en diferentes metodologías, los más conocidos son:

- Filtro Madgwick: Emplea el método de descenso por gradiente para determinar la dirección del error en la medición del giroscopio [36].
- Filtro Mahony: Emplea un filtro paso-bajo a las estimaciones del acelerómetro y un filtro paso-alto a las estimaciones del giroscopio, para posteriormente fusionar ambas estimaciones [37].

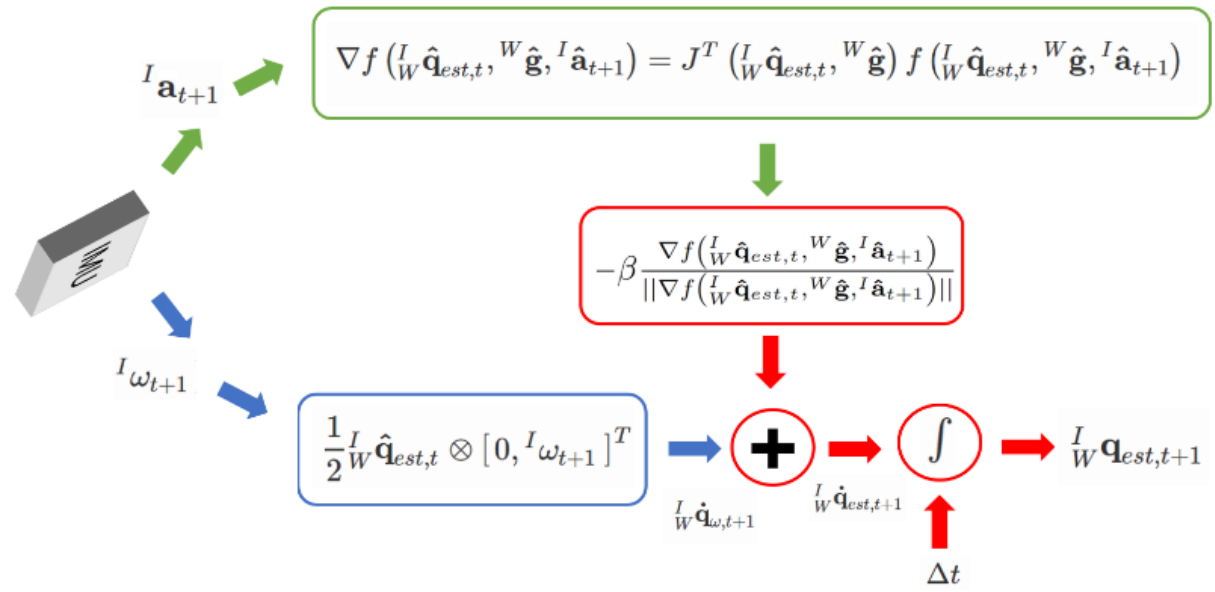


Figura 23: Diagrama de bloques del filtro de Madgwick [36].

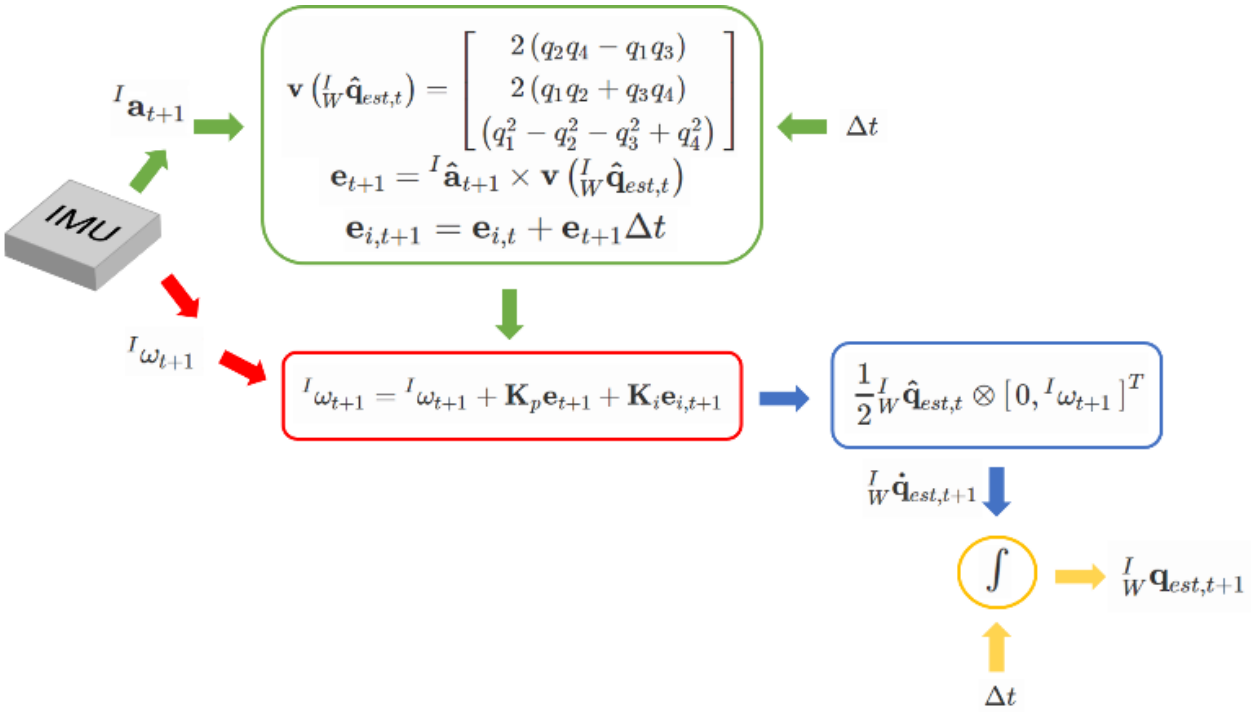


Figura 24: Diagrama de bloques del filtro de Mahony [37].

La desventaja de estos filtros es que se deben de obtener el valor de sus constantes de forma experimental. Es importante tomar en cuenta que, incluso si se trata del mismo sensor, puede tener constantes ligeramente diferentes debido a las variaciones inherentes entre ellos.

Para ahorrar tiempo en programación y calibración, las empresas han desarrollado paquetes de software capaces de procesar los datos de sus sensores con una sola función. Además, estas soluciones incorporan

herramientas de inteligencia artificial para su implementación en proyectos de automatización, como NeuraSense™ de 221e.

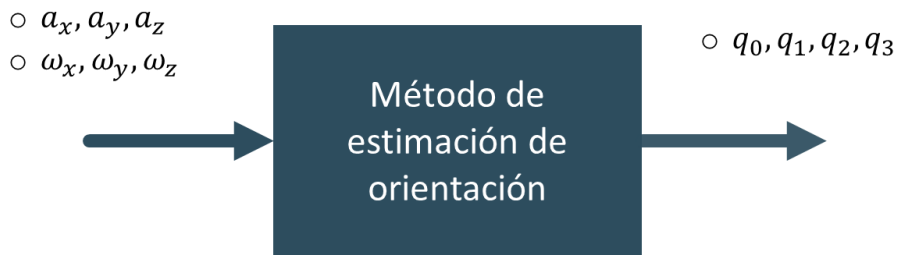


Figura 25: Modelo técnico 2 - Método de estimación de orientación.

Asimismo, muchas empresas han optado por incorporar procesadores integrados en las IMUs para realizar el cálculo de la orientación angular y filtrar los resultados de aceleración y velocidad angular. Algunos ejemplos de estos procesadores son:

- Digital Motion Processor (DMP™) de TDK
- Advanced Pedometer and Event Detection (APEX) de TDK
- Intelligent Sensor Processing Unit (ISPU) de STMicroelectronics

Las alternativas desarrolladas por las empresas suelen superar a los filtros complementarios, ya que estas compañías diseñan estos recursos específicamente para sus productos, teniendo un conocimiento más profundo de estas tecnologías que los usuarios.

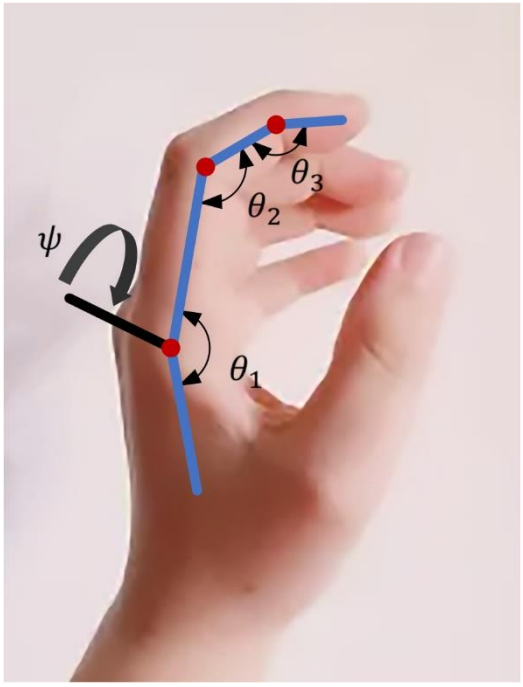


Figura 26: Ángulos variables propuestos para los dedos.

Una vez obtenidos los cuaterniones de orientación, es necesario convertir estos datos a un formato más compacto, unas alternativas serían los ángulos de Euler, pero el único inconveniente es que, al convertir los datos, reducimos en 1 el número de variables de salida, por lo que no es muy factible hacerlo.

Figura 27: Modelo técnico 3 - Modelo cinemático de la tesis.

Al final se propone con ayuda de cálculos, buscar reducir el número de variables de salida posibles, de tal forma que así evitamos reducir el tamaño del bus de transferencia de datos.

4.2 Modelo cinemático de la mano

Actualmente existen modelos cinemáticos de la mano que tienden a ser artificiales, sin considerar las características biológicas de la misma que nos brindan movimientos y posiciones únicas. Una característica de estos modelos es que consideran que las falanges son coplanares, cuando esto es falso, debido al ángulo oblicuo que existe entre las articulaciones interfalángicas y metacarpofalángicas.

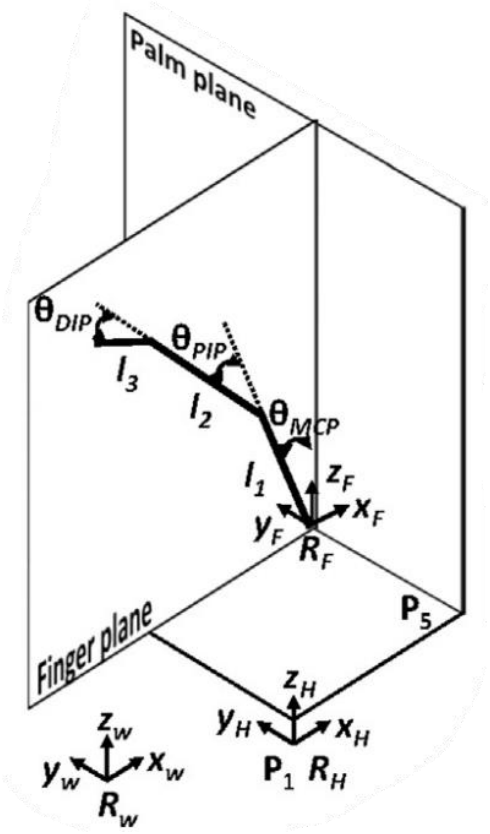


Figura 28: Modelo coplanar del dedo [3].

Esta restricción limita la capacidad de realizar ciertos movimientos y adoptar posiciones específicas con la mano, como la oposición y reposición de los dedos meñique y pulgar. Otra consideración importante en relación con estos modelos es la implementación de softwares capaces de recrear las trayectorias de los dedos basándose en los ángulos de las falanges distales. Esto resulta en una reducción del número de posiciones posibles, lo que, por un lado, mejora la precisión, al tener datos continuos, pero por otro, disminuye la exactitud en la representación de la mano.

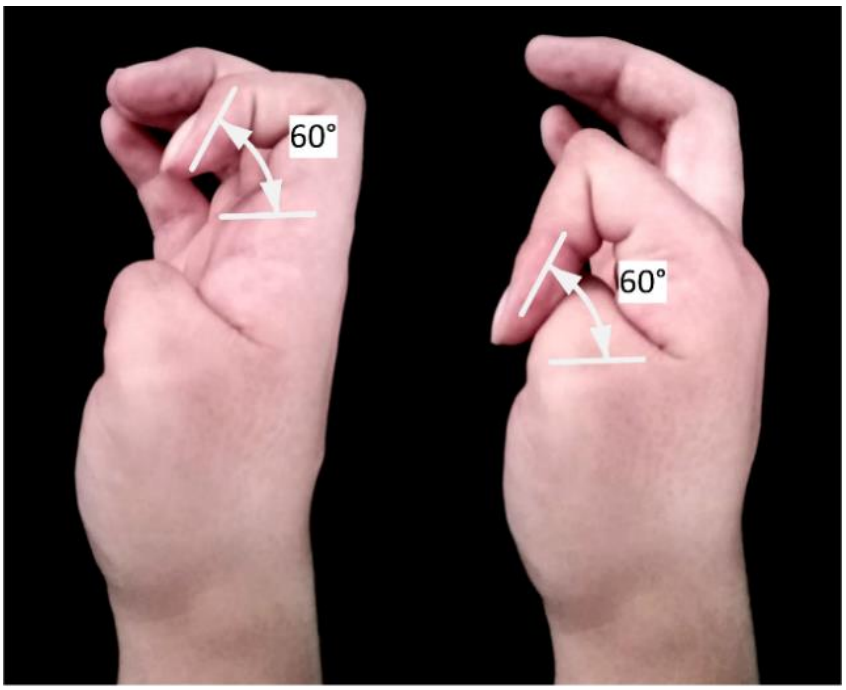


Figura 29: Orientaciones similares en posiciones diferentes de la falange distal del dedo índice de la mano derecha.

En primer lugar, es necesario establecer un sistema de referencia inercial para ubicar la mano en el espacio, este sistema de referencia será la palma de la mano.

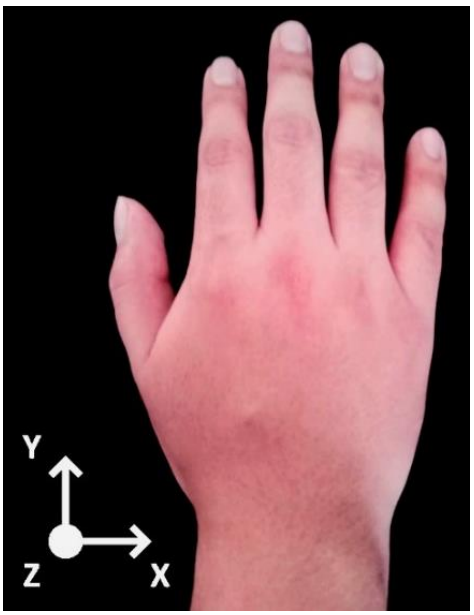


Figura 30: Sistema de referencia de la mano.

Las IMU se colocarán en la parte dorsal de la mano con el propósito de evitar restricciones de movimiento en las falanges al flexionar y

extender los dedos. A su vez, esto provoca que los sensores estén casi en la misma orientación que el sistema de referencia inercial con los dedos extendidos, solo con ligeras variaciones, dependiendo el grosor de los dedos del usuario.

Al tener todos los sensores bajo mismo sistema de referencia inercial, se generan rotaciones resultantes muy complejas, lo que conlleva a un mayor tiempo de procesamiento al tener que realizar más operaciones. Por esta razón, resulta necesario trasladar cada sensor a un sistema de referencia local.

IMU de la Falange Proximal

$$\hat{q}_2^0 = \hat{q}_1^0 \cdot \hat{q}_2^1$$

$$\hat{q}_2^1 = \overline{\hat{q}_1^0} \cdot \hat{q}_2^0 \qquad \textbf{(4.1)}$$

IMU de la Falange Medial

$$\hat{q}_3^0 = \hat{q}_1^0 \cdot \hat{q}_2^1 \cdot \hat{q}_3^2$$

$$\hat{q}_3^2 = \overline{\hat{q}_2^0} \cdot \hat{q}_3^0 \qquad \textbf{(4.2)}$$

IMU de la Falange Distal

$$\hat{q}_3^0 = \hat{q}_1^0 \cdot \hat{q}_2^1 \cdot \hat{q}_3^2 \cdot \hat{q}_4^3$$

$$\hat{q}_4^3 = \overline{\hat{q}_3^0} \cdot \hat{q}_4^0 \qquad \textbf{(4.3)}$$

En donde:

$\hat{q}_1^0$ = Cuaternión unitario inercial de la palma de mano
$\hat{q}_2^0$ = Cuaternión unitario inercial de la falange proximal
$\hat{q}_3^0$ = Cuaternión unitario inercial de la falange medial
$\hat{q}_4^0$ = Cuaternión unitario inercial de la falange distal
$\hat{q}_2^1$ = Cuaternión unitario local de la falange proximal
$\hat{q}_3^2$ = Cuaternión unitario local de la falange medial
$\hat{q}_4^3$ = Cuaternión unitario local de la falange distal

Cada falange cuenta con un grado de libertad, a excepción de la falange proximal, que tiene 2 grados de libertad. Esto implica que cada sensor, ubicado en cada falange, solo representa un dato. Si convertimos los datos a ángulos de Euler, obtendríamos un exceso de información que no sería de mucha utilidad. Dado este escenario y considerando que los ángulos de flexión/extensión rotan a través de un eje oblicuo, se propone crear un modelo cinemático basado en estas premisas.

Empezamos estableciendo las variables y las constantes conocidas:

$\alpha, \beta, \gamma =$ Angulos de Euler de la palma de la mano (variable)
$\psi =$ Ángulo de aducción/abducción metacarpofalangica (variable)
$\theta_1 =$ Ángulo de flexión/extensión metacarpofalangica (variable)
$\theta_2 =$ Ángulo de flexión/extensión interfalangica medial (variable)
$\theta_3 =$ Ángulo de flexión/extensión interfalangica distal (variable)
$\phi_1 =$ Ángulo oblicuo metacarpofalangica (constante)
$\phi_2 =$ Ángulo oblicuo interfalangico medial (constante)
$\phi_3 =$ Ángulo oblicuo interfalangico distal (constante)

Para crear un modelo cinemático, comenzamos con el dedo índice, teniendo en cuenta que el pulgar puede variar en el modelo. Luego, podemos realizar los cálculos utilizando matrices de rotación o cuaterniones. Cada una de estas herramientas tiene sus ventajas y desventajas.

MATRICES DE ROTACIÓN

IMU de la palma de la mano

$$ {}^{0}_{1}\boldsymbol{R} = \boldsymbol{R_x}(\gamma) \cdot \boldsymbol{R_y}(\beta) \cdot \boldsymbol{R_z}(\alpha) $$

$$ {}^{0}_{1}\boldsymbol{R} = \begin{pmatrix} \mathrm{s}(\alpha)\,\mathrm{c}(\beta) & -\mathrm{s}(\alpha)\,\mathrm{c}(\beta) & \mathrm{s}(\beta) \\ \mathrm{c}(\alpha)\,\mathrm{s}(\beta)\,\mathrm{s}(\gamma) + \mathrm{s}(\alpha)\,\mathrm{c}(\gamma) & \mathrm{c}(\alpha)\,\mathrm{c}(\gamma) - \mathrm{s}(\alpha)\,\mathrm{s}(\beta)\,\mathrm{s}(\gamma) & -\mathrm{c}(\beta)\,\mathrm{s}(\gamma) \\ \mathrm{s}(\alpha)\,\mathrm{s}(\gamma) - \mathrm{c}(\alpha)\,\mathrm{s}(\beta)\,\mathrm{c}(\gamma) & \mathrm{s}(\alpha)\,\mathrm{s}(\beta)\,\mathrm{c}(\gamma) + \mathrm{c}(\alpha)\,\mathrm{s}(\gamma) & \mathrm{c}(\beta)\,\mathrm{c}(\gamma) \end{pmatrix} \quad \textbf{(4.4)} $$

IMU de la Falange Proximal

$$ {}^{1}_{2}\boldsymbol{R} = \boldsymbol{R_z}(\phi_1) \cdot \boldsymbol{R_x}(-\theta_1) \cdot \boldsymbol{R_z}(\psi) \cdot \boldsymbol{R_z}(-\phi_1) = \boldsymbol{R_z}(\phi_1) \cdot \boldsymbol{R_x}(-\theta_1) \cdot \boldsymbol{R_z}(\psi - \phi_1) $$

$$ {}^{1}_{2}\boldsymbol{R} = \begin{pmatrix} \mathrm{c}(\phi_1)\,\mathrm{c}(\psi - \phi_1) - \mathrm{c}(\theta_1)\,\mathrm{s}(\phi_1)\,\mathrm{s}(\psi - \phi_1) & -\mathrm{c}(\theta_1)\,\mathrm{s}(\phi_1)\,\mathrm{c}(\psi - \phi_1) - \mathrm{c}(\phi_1)\,\mathrm{s}(\psi - \phi_1) & -\mathrm{s}(\theta_1)\,\mathrm{s}(\phi_1) \\ \mathrm{c}(\theta_1)\,\mathrm{c}(\phi_1)\,\mathrm{s}(\psi - \phi_1) + \mathrm{s}(\phi_1)\,\mathrm{c}(\psi - \phi_1) & \mathrm{c}(\theta_1)\,\mathrm{c}(\phi_1)\,\mathrm{c}(\psi - \phi_1) - \mathrm{s}(\phi_1)\,\mathrm{s}(\psi - \phi_1) & \mathrm{s}(\theta_1)\,\mathrm{c}(\phi_1) \\ -\mathrm{s}(\theta_1)\,\mathrm{s}(\psi - \phi_1) & -\mathrm{s}(\theta_1)\,\mathrm{c}(\psi - \phi_1) & \mathrm{c}(\theta_1) \end{pmatrix} \quad \textbf{(4.5)} $$

IMU de la Falange Medial

$$ {}^{2}_{3}\boldsymbol{R} = \boldsymbol{R_z}(\phi_2) \cdot \boldsymbol{R_x}(-\theta_2) \cdot \boldsymbol{R_z}(-\phi_2) $$

$$ {}^{2}_{3}\boldsymbol{R} = \begin{pmatrix} \mathrm{c}(\theta_2)\,\mathrm{s}^2(\phi_2) + \mathrm{c}^2(\phi_2) & \mathrm{s}^2\left(\frac{\theta_2}{2}\right)\mathrm{s}(2\phi_2) & -\mathrm{s}(\theta_2)\,\mathrm{s}(\phi_2) \\ \mathrm{s}^2\left(\frac{\theta_2}{2}\right)\mathrm{s}(2\phi_2) & \mathrm{c}(\theta_2)\,\mathrm{c}^2(\phi_2) + \mathrm{s}^2(\phi_2) & \mathrm{s}(\theta_2)\,\mathrm{c}(\phi_2) \\ \mathrm{s}(\theta_2)\,\mathrm{s}(\phi_2) & -\mathrm{s}(\theta_2)\,\mathrm{c}(\phi_2) & \mathrm{c}(\theta_2) \end{pmatrix} \quad \textbf{(4.6)} $$

IMU de la Falange Distal

$$ {}^{3}_{4}\boldsymbol{R} = \boldsymbol{R_z}(\phi_3) \cdot \boldsymbol{R_x}(-\theta_3) \cdot \boldsymbol{R_z}(-\phi_3) $$

$$ {}^{3}_{4}\boldsymbol{R} = \begin{pmatrix} \mathrm{c}(\theta_3)\,\mathrm{s}^2(\phi_3) + \mathrm{c}^2(\phi_3) & \mathrm{s}^2\left(\frac{\theta_3}{2}\right)\mathrm{s}(2\phi_3) & -\mathrm{s}(\theta_3)\,\mathrm{s}(\phi_3) \\ \mathrm{s}^2\left(\frac{\theta_3}{2}\right)\mathrm{s}(2\phi_3) & \mathrm{c}(\theta_3)\,\mathrm{c}^2(\phi_3) + \mathrm{s}^2(\phi_3) & \mathrm{s}(\theta_3)\,\mathrm{c}(\phi_3) \\ \mathrm{s}(\theta_3)\,\mathrm{s}(\phi_3) & -\mathrm{s}(\theta_3)\,\mathrm{c}(\phi_3) & \mathrm{c}(\theta_3) \end{pmatrix} \quad \textbf{(4.7)} $$

Partiendo de los cuaterniones proporcionados por los sensores, es necesario incorporarlos en la fórmula (3.8). Luego, igualamos la matriz con las ecuaciones (4.5), (4.6) y (4.7), por último, procedemos a despejar las variables deseadas, en este proceso, se busca que el resultado este en función de ArcTan debido a que el resultado de Arctan2 se puede representar en cualquiera de los cuatro cuadrantes. Una vez despejadas las variables, obtenemos como las ecuaciones resultantes:

$$\psi = \tan^{-1}\left(\frac{q_1 q_3 - q_0 q_2}{q_0 q_1 + q_2 q_3}\right) + \phi_1 \tag{4.8}$$

$$\theta_1 = -\tan^{-1}\left(\frac{2(q_0 q_1 - q_2 q_3)}{(2q_0^2 + 2q_3^2 - 1)\cos(\phi_1)}\right) \tag{4.9}$$

$$\theta_2 = -\tan^{-1}\left(\frac{2(q_0 q_1 - q_2 q_3)}{(2q_0^2 + 2q_3^2 - 1)\cos(\phi_2)}\right) \tag{4.10}$$

$$\theta_3 = -\tan^{-1}\left(\frac{2(q_0 q_1 - q_2 q_3)}{(2q_0^2 + 2q_3^2 - 1)\cos(\phi_3)}\right) \tag{4.11}$$

Ventajas:

- Las matrices consideran todos los datos del cuaternión, lo que resulta en una mayor precisión al no excluir ningún dato de este.
- Las matrices son más fáciles de trabajar y de visualizar en el espacio 3D.

Desventajas:

- Las operaciones son más extensas.
- Las matrices tienen singularidades, esto debido que en los dos primeros valores de la diagonal es imposible obtener un 0, ya que ϕ jamás va es 0°.

CUATERNIONES

En el caso de los cuaterniones, el planteamiento es el mismo, pero en lugar de utilizar matrices de rotación, empleamos cuaterniones.

Cuaternión de rotación de la Falange Proximal

$$\hat{q}_2^1 = q_z(\phi_1)\cdot q_x(-\theta_1)\cdot q_z(\psi)\cdot q_z(-\phi_1) = q_z(\phi_1)\cdot q_x(-\theta_1)\cdot q_z(\psi-\phi_1)$$

$$\hat{q}_2^1 = \begin{pmatrix} \cos\left(\frac{\theta_1}{2}\right)\cos\left(\frac{\psi}{2}\right) \\ -\sin\left(\frac{\theta_1}{2}\right)\cos\left(\frac{1}{2}(\psi-2\phi_1)\right) \\ \sin\left(\frac{\theta_1}{2}\right)\sin\left(\frac{1}{2}(\psi-2\phi_1)\right) \\ \cos\left(\frac{\theta_1}{2}\right)\sin\left(\frac{\psi}{2}\right) \end{pmatrix} \tag{4.12}$$

Cuaternión de rotación de la Falange Medial

$$\hat{q}_3^2 = q_z(\phi_2) \cdot q_x(-\theta_2) \cdot q_z(-\phi_2)$$

$$\hat{q}_3^2 = \begin{pmatrix} \cos\left(\frac{\theta_2}{2}\right) \\ -\sin\left(\frac{\theta_2}{2}\right)\cos(\phi_2) \\ -\sin\left(\frac{\theta_2}{2}\right)\sin(\phi_2) \\ 0 \end{pmatrix} \quad \textbf{(4.13)}$$

Cuaternión de rotación de la Falange Distal

$$\hat{q}_4^3 = q_z(\phi_3) \cdot q_x(-\theta_3) \cdot q_z(-\phi_3)$$

$$\hat{q}_4^3 = \begin{pmatrix} \cos\left(\frac{\theta_3}{2}\right) \\ -\sin\left(\frac{\theta_3}{2}\right)\cos(\phi_3) \\ -\sin\left(\frac{\theta_3}{2}\right)\sin(\phi_3) \\ 0 \end{pmatrix} \quad \textbf{(4.14)}$$

Despejamos las variables buscadas de las ecuaciones (4.12), (4.13) y (4.14), dando como resultado las ecuaciones (4.15), (4.16), (4.17) y (4.18):

$$\psi = 2\tan^{-1}\left(\frac{q_3}{q_0}\right) \quad \textbf{(4.15)}$$

$$\theta_1 = -2\tan^{-1}\left(\frac{q_1}{q_3\sin(\phi_1) + q_0\cos(\phi_1)}\right) \quad \textbf{(4.16)}$$

$$\boldsymbol{\theta_2 = -2\tan^{-1}\left(\frac{q_1}{q_0\cos(\phi_2)}\right)} \quad \textbf{(4.17)}$$

$$\theta_3 = -2\tan^{-1}\left(\frac{q_1}{q_0\cos(\phi_3)}\right) \quad \textbf{(4.18)}$$

Ventajas:

- Los cuaterniones son un conjunto de datos más compactos a comparación de las matrices de rotación.
- Las ecuaciones resultantes son más pequeñas
- Las ecuaciones no cuentan con singularidades en ningún punto

Desventajas:

- Las rotaciones no consideran muchos datos, por lo que el sistema este forzado a tener las constantes del modelo lo más cercanas a la realidad posibles.
- Es necesario trabajar con todos los valores del cuaternión, principalmente con la parte real.

Según se puede apreciar, la diferencia entre ambos modelos es significativa, pero su desempeño depende en gran medida de las características del sensor y del sistema. Si el sensor está mal orientado,

el modelo de cuaterniones tendrá un error más considerable en comparación con las matrices de rotación. Por otro lado, si el sistema abarca rangos que superan los tres cuadrantes, las matrices de rotación presentarán más singularidades y, en consecuencia, un mayor número de errores. En última instancia, ambos inconvenientes pueden estar presentes tanto en los sensores como en el sistema, por lo que ambos modelos son igualmente válidos.

Para verificar la confiabilidad del modelo del dedo, se creó una simulación de un dedo genérico en la que es posible modificar los valores de los ángulos oblicuos para visualizar las posiciones y movimientos posibles del dedo.

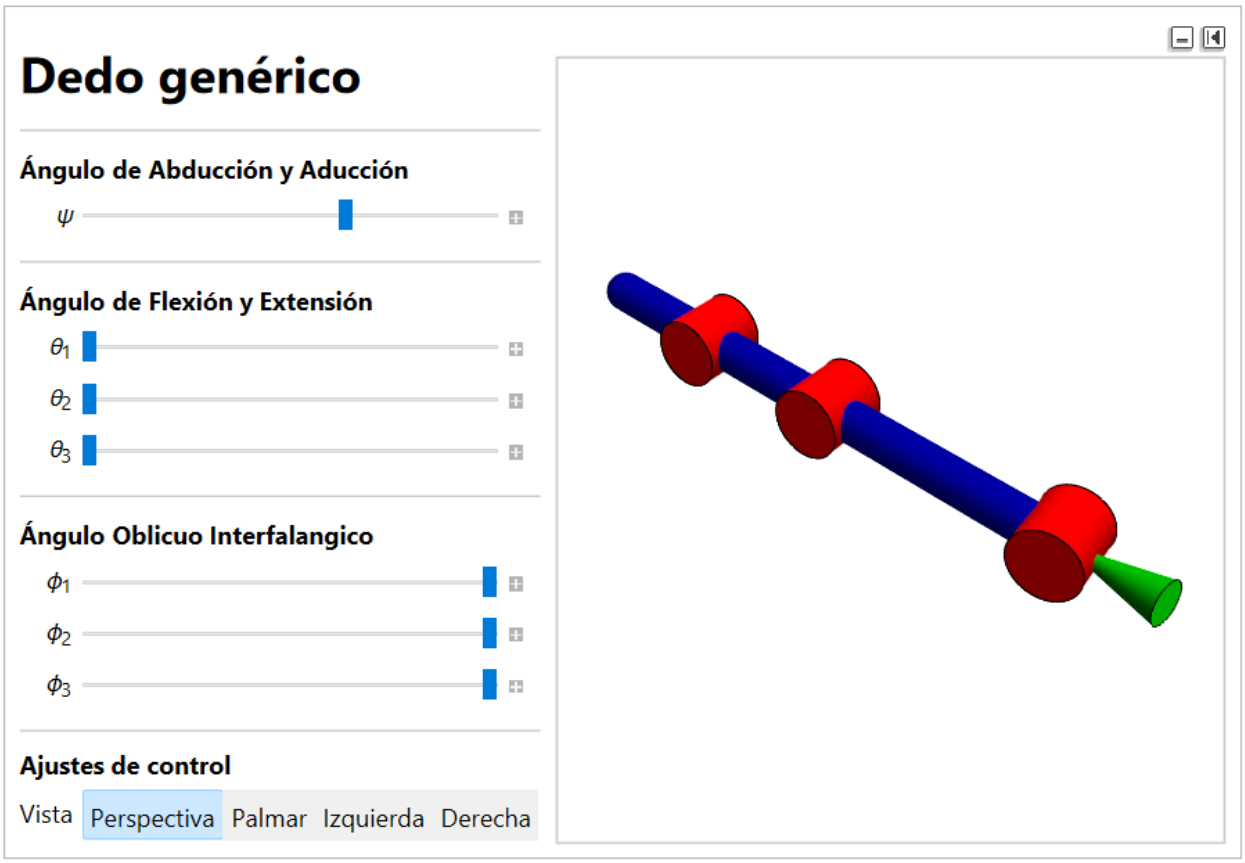


Figura 31: Captura del programa que simula el movimiento de un dedo.

Con ayuda de esta simulación, se logró visualizar un movimiento diferente a los planteados por otros modelos, en donde el dedo tiene un movimiento tridimensional a pesar de tener 1 grado de libertad por cada falange, ampliando el rango de alcance del dedo simulado.

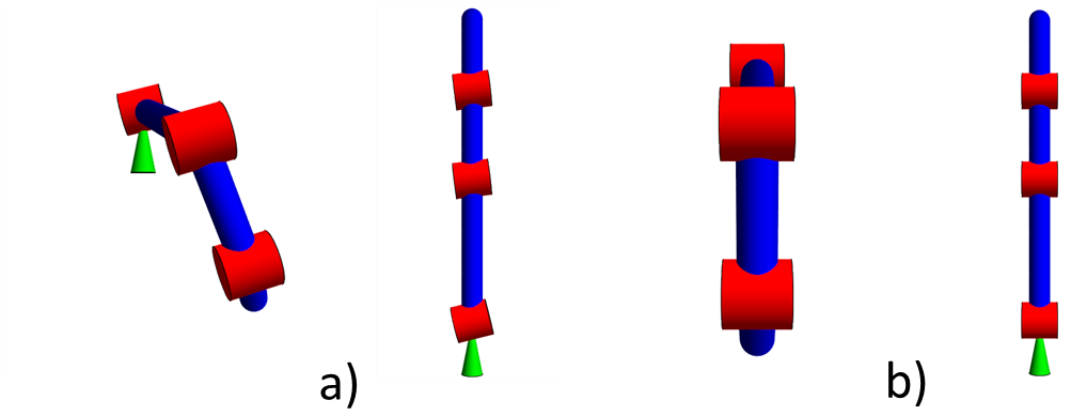


Figura 32: Diferencia entre el modelo coplanar y no coplanar.

Revisando las posiciones posibles, se puede suponer que, si ajustamos los valores de los ángulos oblicuos a valores grandes, obtendríamos el mismo efecto del movimiento oposición y reposición de dedo meñique.

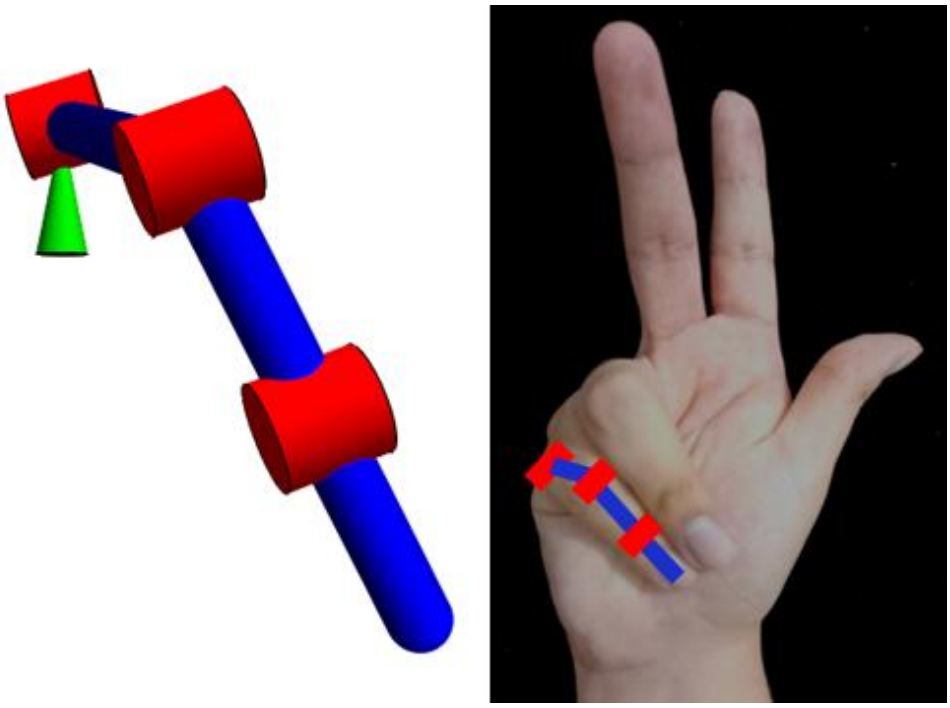


Figura 33: Comparación de la simulación con una fotografía del dedo meñique.

4.3 Diseño del modelo funcional del guante

Para el modelo funcional se utilizará el dedo índice y la palma de la mano para verificar las ecuaciones que nos proporciona el modelo cinemático, en donde los ángulos interfalángicos constantes son con base en la Figura 30, en la que se pueden visualizar los ángulos oblicuos interfalángicos.

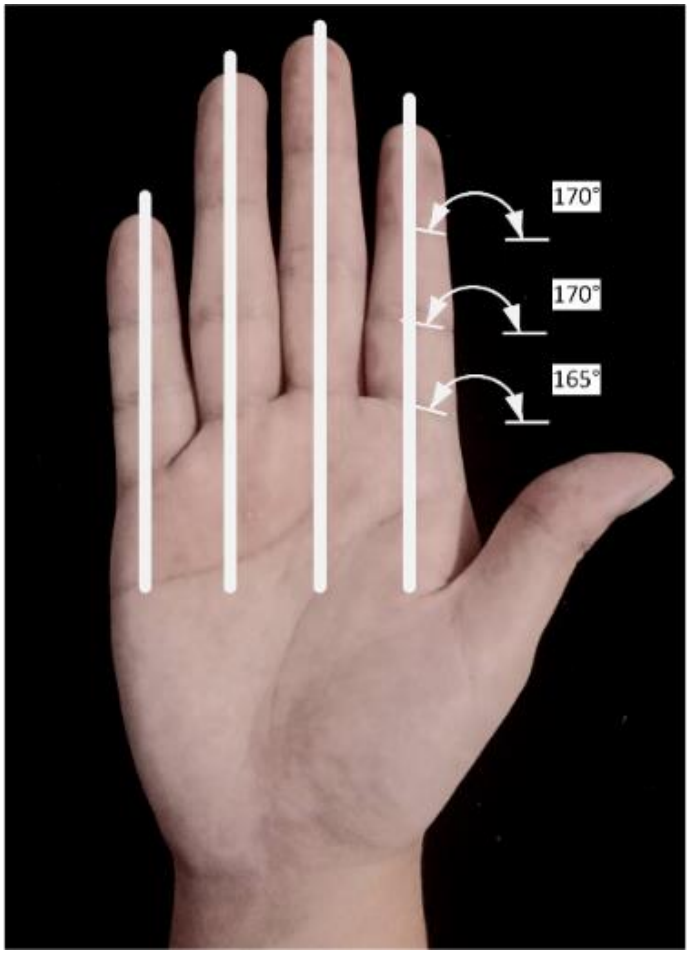


Figura 34: Imagen de referencia de los ángulos oblicuos del dedo índice.

SUBSISTEMA: HARDWARE DEL MODELO FUNCIONAL

El modelo funcional capturará únicamente el movimiento de un dedo, por lo que requerimos solo 4 sensores: uno para cada falange y uno para la palma. Dado que utilizamos sensores idénticos, todos comparten la misma dirección I2C, lo que podría causar confusión al adquirir los datos de cada uno. Por lo tanto, es esencial utilizar un multiplexor I2C. Además, necesitamos un microcontrolador para gestionar los sensores, realizar cálculos y conversiones.

Para el diseño del modelo funcional se buscaron componentes que fueron de fácil acceso y de bajo costo para realizar pruebas rápidas y validar el modelo cinemático propuesto. Los componentes utilizados en el modelo funcional son:

- IMU MPU-6050 x4
- Multiplexor I2C TCA9548A
- Microcontrolador ESP32 WROOM de 16 MB
- Protoboard de 300 puntos
- Cables para protoboard de 20 cm

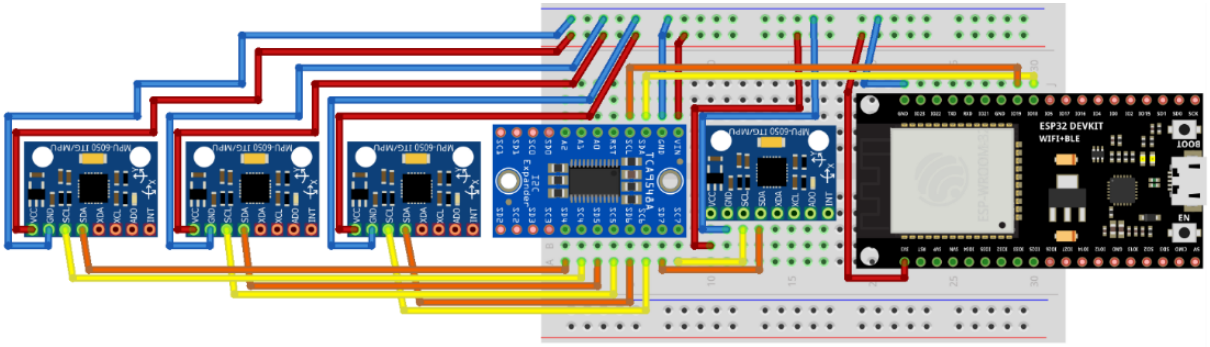
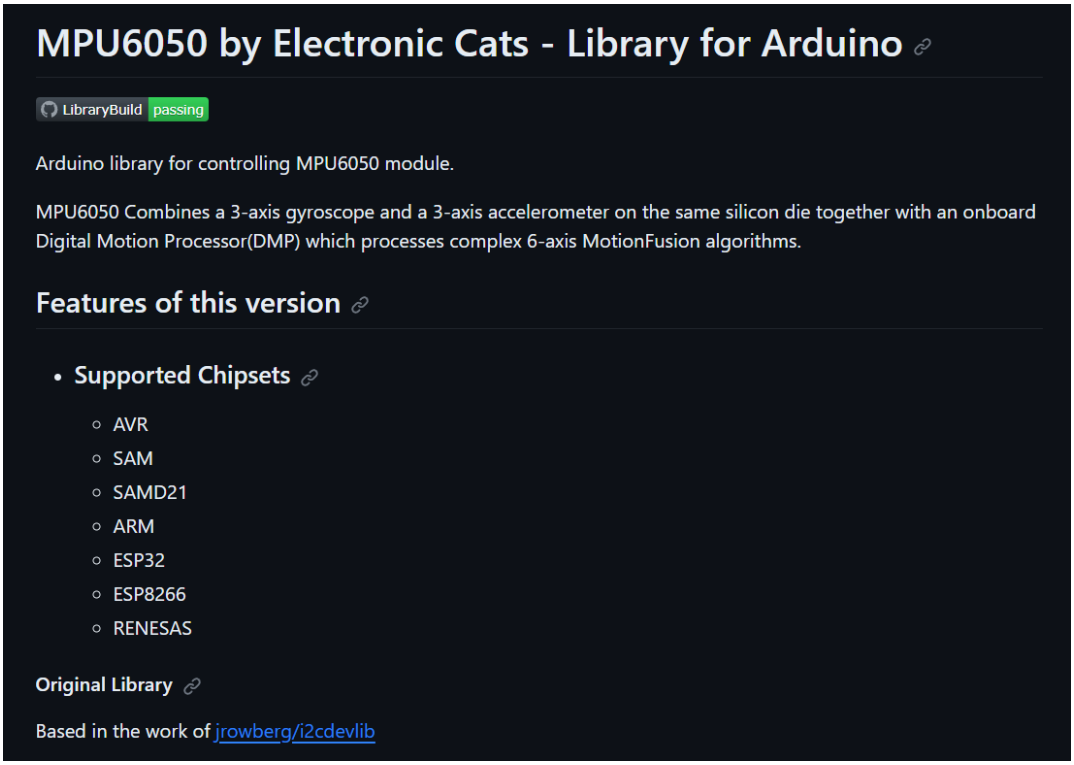


Figura 35: Diagrama de conexiones del modelo funcional.

El MPU-6050 es un IMU que incorpora un procesador integrado DMP para realizar cálculos complejos. Además, dispone de un bus I2C auxiliar que permite acceder a un magnetómetro externo. Aunque este sensor ya no se recomienda para nuevos diseños, actualmente es el más económico y fácil de encontrar en el mercado.

Figura 36: Esquemático de la serie MPU-6000 [42].

Para acceder a las funciones del DMP, se utilizará una biblioteca desarrollada por Electronic Cats. Esta biblioteca no solo proporciona acceso al DMP, sino que también incluye funciones para calcular ángulos de Euler, aceleración lineal y velocidad angular, que ya han sido filtrados. Además, ofrece un controlador PID para calibrar el acelerómetro y el giroscopio del sensor.



MPU6050 by Electronic Cats - Library for Arduino

LibraryBuild passing

Arduino library for controlling MPU6050 module.

MPU6050 Combines a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die together with an onboard Digital Motion Processor(DMP) which processes complex 6-axis MotionFusion algorithms.

Features of this version

- **Supported Chipsets**
  - AVR
  - SAM
  - SAMD21
  - ARM
  - ESP32
  - ESP8266
  - RENESAS

Original Library

Based in the work of jrowberg/i2cdevlib

Figura 37: Captura de la biblioteca del MPU-6050 [10].

El multiplexor I2C TCA9548A tiene la facilidad de que podemos cambiar el selector vía código, sin la necesidad de usar más pines del microcontrolador como en los multiplexores tradicionales.

```
// Uso de TCA9548A
void TCA9548A(uint8_t bus){
  Wire.beginTransmission(0x70); // TCA9548A dirección es 0x70
  Wire.write(1 << bus);         // Buscamos el byte selector
  Wire.endTransmission();       // Cerramos transmisión
}
```

Figura 38: Captura del código del TCA9548A [33].

Por parte del microcontrolador ESP32, tenemos acceso a una gran variedad de herramientas como nos pueden auxiliar entre todos destacan un procesador de dos núcleos y la conectividad wifi y bluetooth. La ventaja de la conexión wifi es que podemos utilizar la tecnología OTA (Over-The-Air), también conocida como programación inalámbrica.

La tecnología OTA es aquella que utilizan todos los dispositivos móviles para poder actualizar su firmware sin la necesidad de estar conectada por cable a una computadora, esta actualización se descarga por internet y posteriormente se instala el nuevo firmware. El ESP32 cuenta con soporte OTA [17].

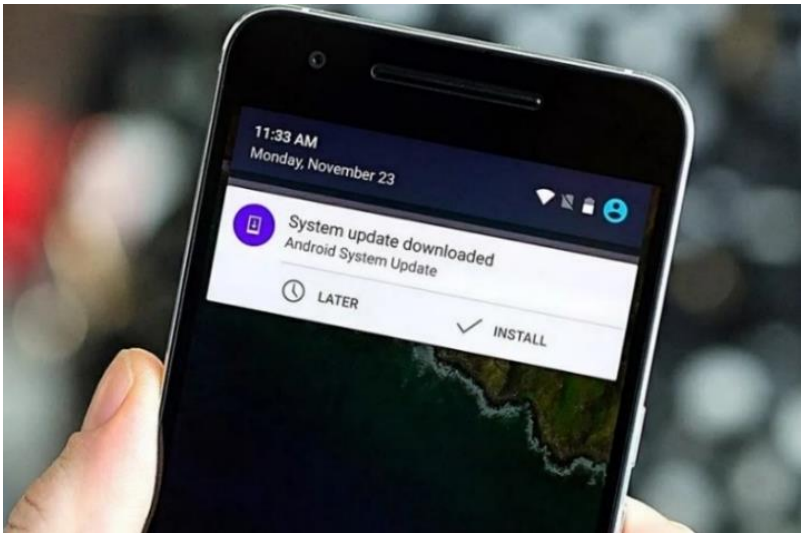


Figura 39: Ejemplo de la tecnología OTA [17].

Una vez que hemos identificado los componentes y su importancia en el modelo funcional, la siguiente fase consiste en adherir el circuito al guante. Es crucial resaltar que las IMUs se fijan en cada falange del dedo índice, y estos sensores se conectan mediante cables jumper de 20 cm.

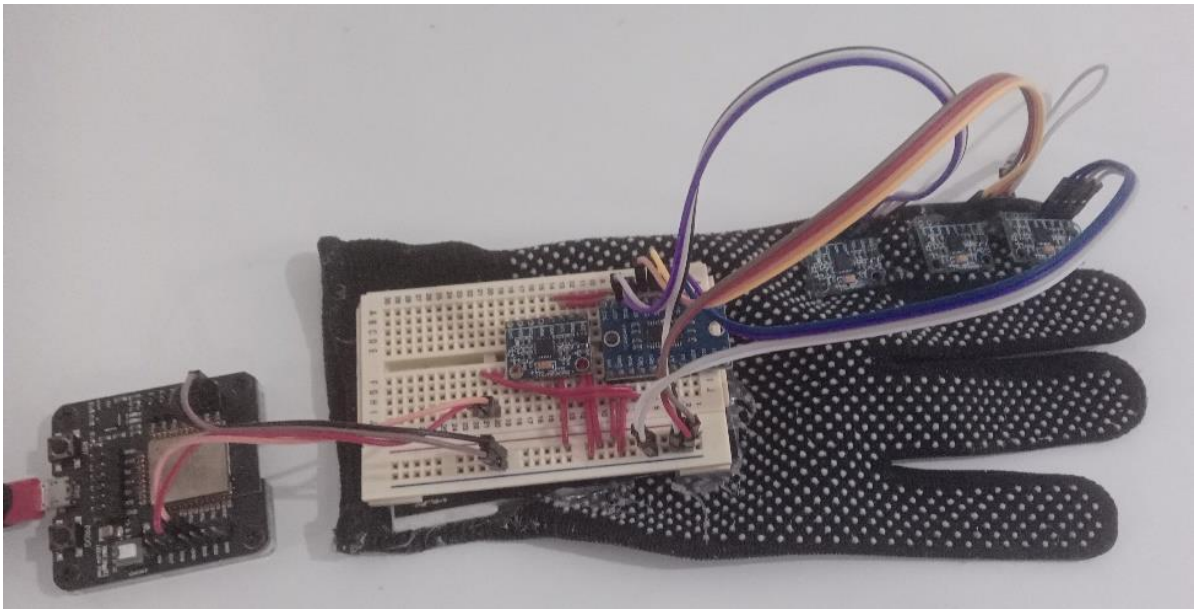


Figura 40: Fotografía del modelo funcional.

SUBSISTEMA: SOFTWARE DEL MODELO FUNCIONAL

El algoritmo por utilizar se basa en el uso del núcleo Dual-Core del ESP32, logrando desarrollar programas multitareas para lograr un mejor

rendimiento en nuestros programas, para eso se prestablece que acciones le tocaran a cada núcleo:

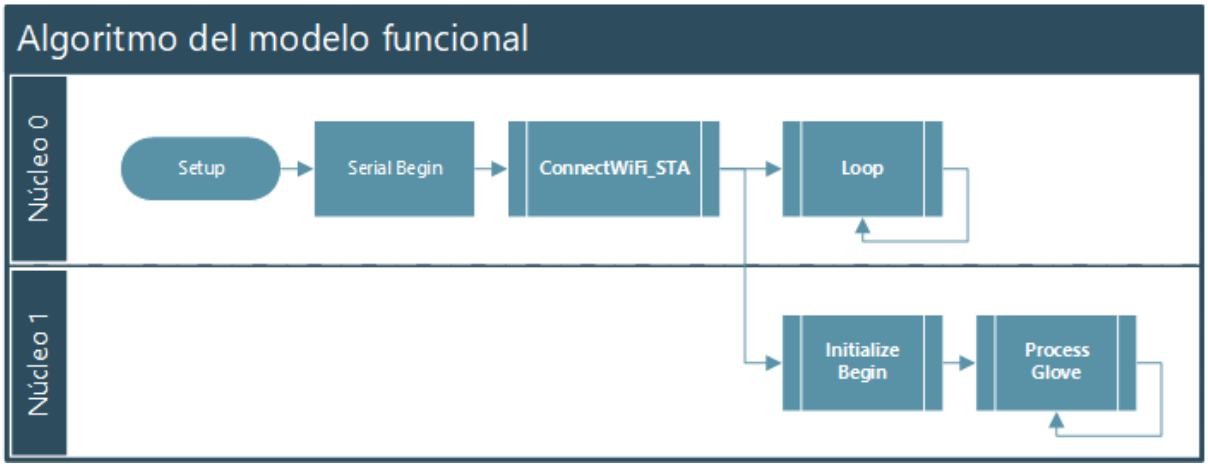


Figura 41: Diagrama de flujo del modelo funcional

Los bloques más importantes del diagrama son todos aquellos que se encuentran en el núcleo 1, debido a que el núcleo 0 se encarga la conexión OTA con el modem. Dicho código es prestablecido por defecto por lo que no es importante destacar la inicialización del wifi, bluetooth y OTA.

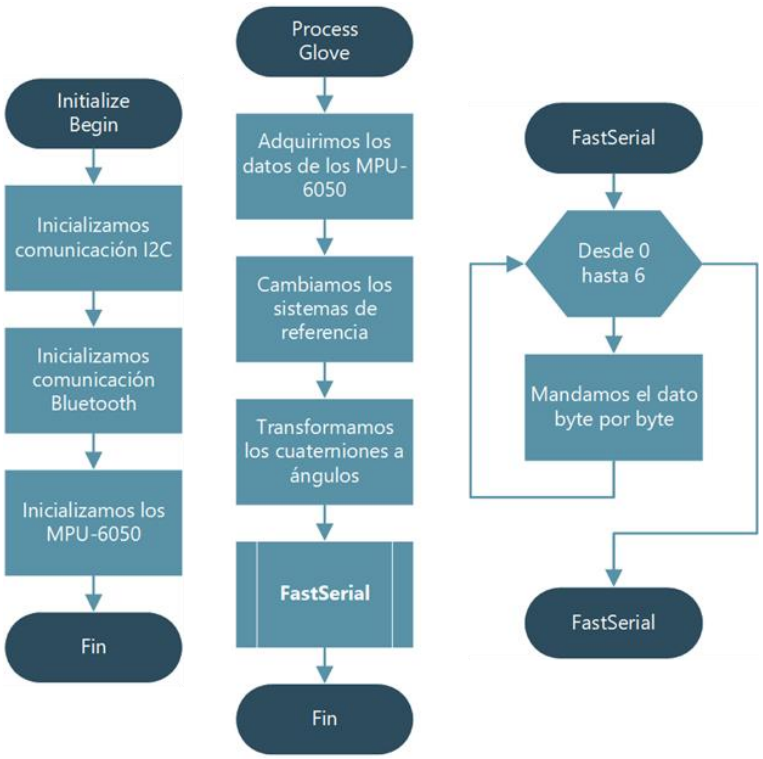


Figura 42: Diagrama de flujo de los subprocesos del núcleo 1.

Para mejorar la transferencia de datos, se ha implementado un sistema que reduce significativamente el tiempo de transferencia al disminuir la

cantidad de bytes enviados a través de la comunicación serial. Inicialmente, los datos se transmiten carácter por carácter utilizando el tipo de dato *'char'*. Sin embargo, al enviar valores con decimales, se reduce la precisión a dos decimales significativos. Con este primer algoritmo, la velocidad de transmisión de datos puede variar entre 5 a 8 bytes, dependiendo de la información proporcionada por los sensores y el proceso de conversión de datos.

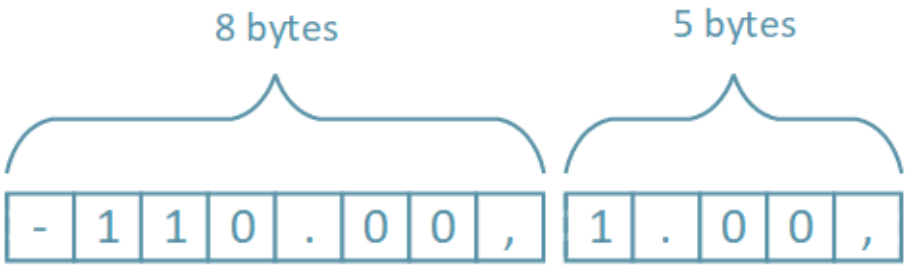


Figura 43: Tamaño en bytes de cada dato (algoritmo 1).

Para este nuevo sistema transformamos el tipo de dato por defecto *'float'* a tipo de dato *'int'* (entero) para reducir el número de bytes que se transfieren por la comunicación serial de 4 bytes a 2 bytes, debido a que trabajamos con números decimales y buscamos una resolución de 0.1 [°] de los ángulos obtenidos por el modelo cinemático, multiplicamos el numero decimal por 10, posteriormente el dispositivo receptor tiene que decodificar este tipo de dato dividiéndolo entre 10 para obtener un valor decimal otra vez. A este algoritmo se le denomino con el nombre de ***FastSerial***.

Figura 44: Tamaño en bytes de cada dato (algoritmo 2).

Es importante señalar que en este algoritmo no se utilizan caracteres de salto de línea. Para determinar el inicio del bus de datos, se empleó un identificador de 1 byte con un valor de 208.

SUBSISTEMA: ADQUISICIÓN DE DATOS

Para validar de manera más cualitativa el modelo funcional, se creó un programa en C# con la ayuda del motor gráfico Unity. Se desarrolló un *GameObject* compuesto por *GameObjects* vacíos, que actúan como un sistema de referencia rotacional. Se les agregaron mallas a estos puntos de referencia para darle presentación y forma al modelo del dedo.

El mallado se realizó en Blender®, incorporando un diseño de dedo robótico para visualizar de manera más clara el movimiento de las articulaciones. Esto se hizo con el objetivo de mejorar la representación visual, ya que algunos modelos 3D existentes presentan deformaciones inconsistentes debido a la baja cantidad de cuadriláteros en su estructura de mallado.

En cuanto al movimiento de los dedos, se empleó el modelo cinemático propuesto, pero con la variante de cuaterniones. Esto se debió a que los motores gráficos suelen trabajar con esta técnica, ya que los cálculos son más rápidos. Además, se adaptó el código y las rotaciones al sistema de mano izquierda utilizado por el motor gráfico.

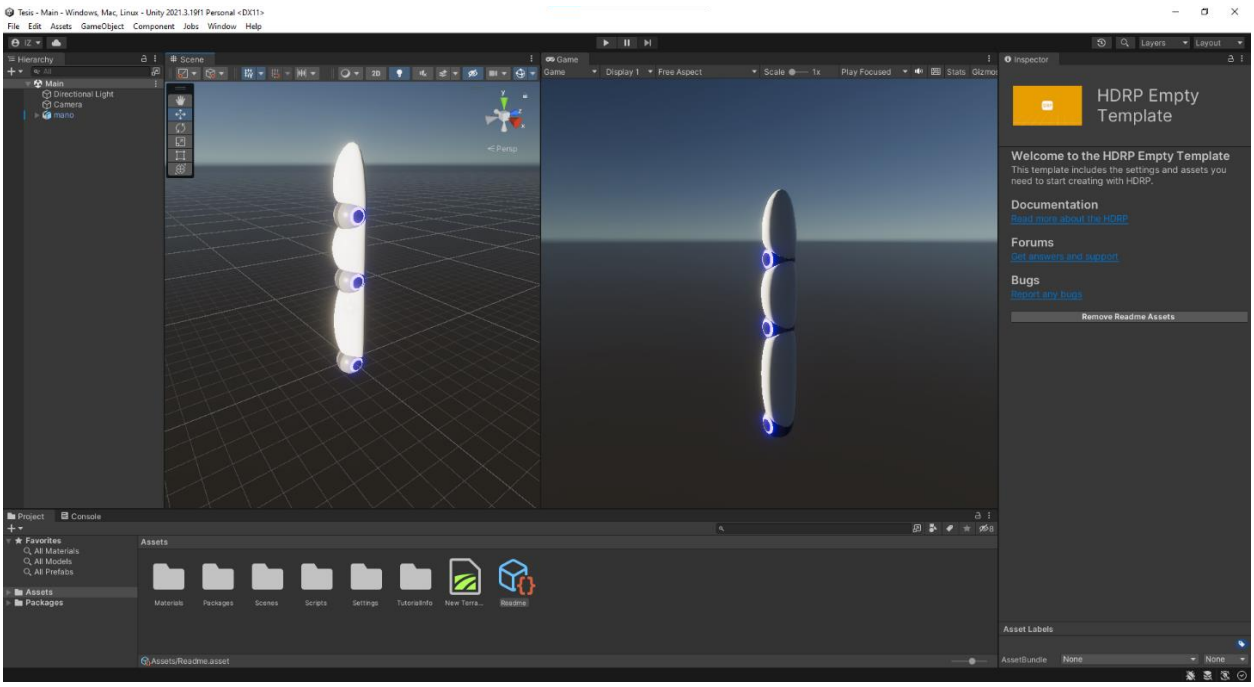


Figura 45: Captura del programa en Unity.

CAPÍTULO 5: PRUEBAS DE PROCESAMIENTO

5.1 Pruebas de procesamiento con el modelo cinemático

Para verificar el tiempo de procesamiento del modelo cinemático y su correspondiente algoritmo, se realizó una sencilla prueba con el modelo funcional, en donde se ejecutaron 2 algoritmos diferentes:

- Algoritmo 1: Consta de mandar los componentes de los cuaterniones de las IMUs sin procesarlos, además de mandar los datos en forma de cadena de caracteres, que es la forma por defecto que nos proporciona el software de Arduino.
    - Cantidad de datos: 12 datos por dedo + 4 datos por la palma
    - Tamaño en bytes por cada dato: 5-8 bytes
    - Velocidad por bits: 115200 [baudios]
    - Iteraciones por segundo: 294 [Hz]
- Algoritmo 2: Consta de procesador los componentes de los cuaternión obtenidos por las IMUs utilizando las ecuaciones resultantes del modelo cinemático planteado con anterioridad, además de codificar los datos en grupos de 2 bytes del tipo de dato *'int16_t'*, posteriormente el receptor decodifica este dato.
    - Cantidad de datos: 4 datos por dedo + 3 datos por la palma
    - Tamaño en bytes por cada dato: 2 bytes
    - Velocidad por bits: 115200 [baudios]
    - Iteraciones por segundo: 101 [Hz]

Al realizar una prueba de tiempos, se obtuvo el número de iteraciones en ambos algoritmos, considerando que se utilizaron solo 4 sensores, a pesar de que el sistema contaría con un total de 16 sensores, tomando en cuenta que el sensor de la palma de la mano es el que envía más datos y requiere más cálculos. Aunque los números de iteraciones obtenidos no son definitivos, es posible suponer una frecuencia de iteración aproximada dividiendo la frecuencia obtenida entre 4.

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
entry 0x400805f0
Connection Success!
>......>......
Connection Success!
>*......>......
Connection Success!
>*......>......
Connection Success!
>*......>......
```

```
Iteraciones por segundo algoritmo nuevo: 294
208,1.00,-0.00,0.00,-0.00,1.00,0.00,0.00,0.00,1.00,0.00,0.00,0.00,1.00,0.
Iteraciones por segundo algoritmo viejo: 101
```

Figura 46: Captura de los resultados de las pruebas de procesamiento

Un detalle importante durante la prueba es que el programa no calcula el tiempo que tarda en ejecutar el algoritmo. Esto se debe a que las funciones “millis” y “micros” han demostrado tener errores al medir el tiempo, debido al uso extensivo de estas funciones en la paquetería de Electronic Cats. Además, estas funciones son muy sensibles y pueden verse fácilmente afectadas por otras sentencias, como las sentencias *If-Else*.

En resumen, se puede ver que hubo un cambio significativo entre los dos algoritmos, mejorando por mucho el tiempo de iteración, por lo que se ve que es viable la técnica utilizada en el algoritmo 2.

5.2 Pruebas con el modelo funcional del guante

Se llevaron a cabo pruebas utilizando el motor gráfico Unity para evaluar de manera empírica la respuesta y la sensibilidad del modelo funcional. Esta elección se debió a la falta de disponibilidad de otro dispositivo capaz de medir de manera cuantitativa y precisa el movimiento de los dedos.

Es relevante destacar que la conexión Bluetooth puede experimentar fallas según el dispositivo al que se conecte. En estas pruebas, se optó por una conexión por cable.

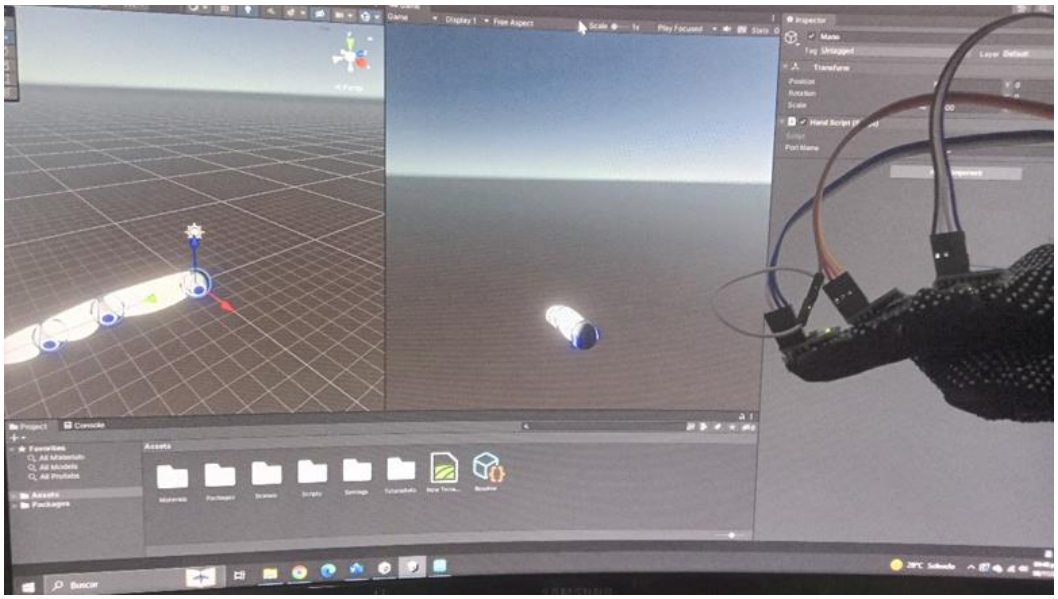


Figura 47: Evidencia 1 de las pruebas con el modelo funcional.

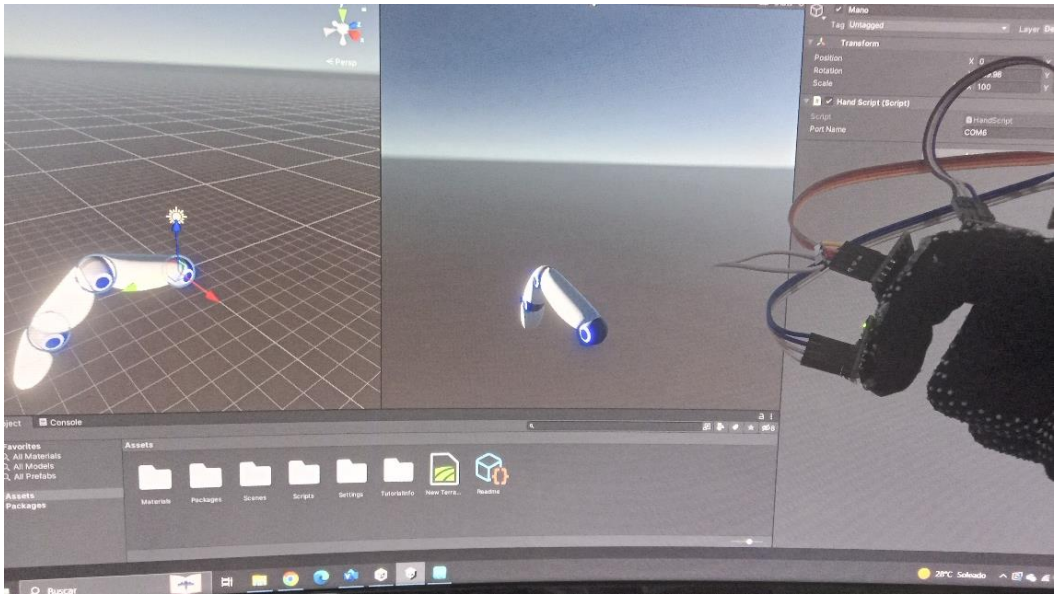


Figura 48: Evidencia 2 de las pruebas con el modelo funcional.

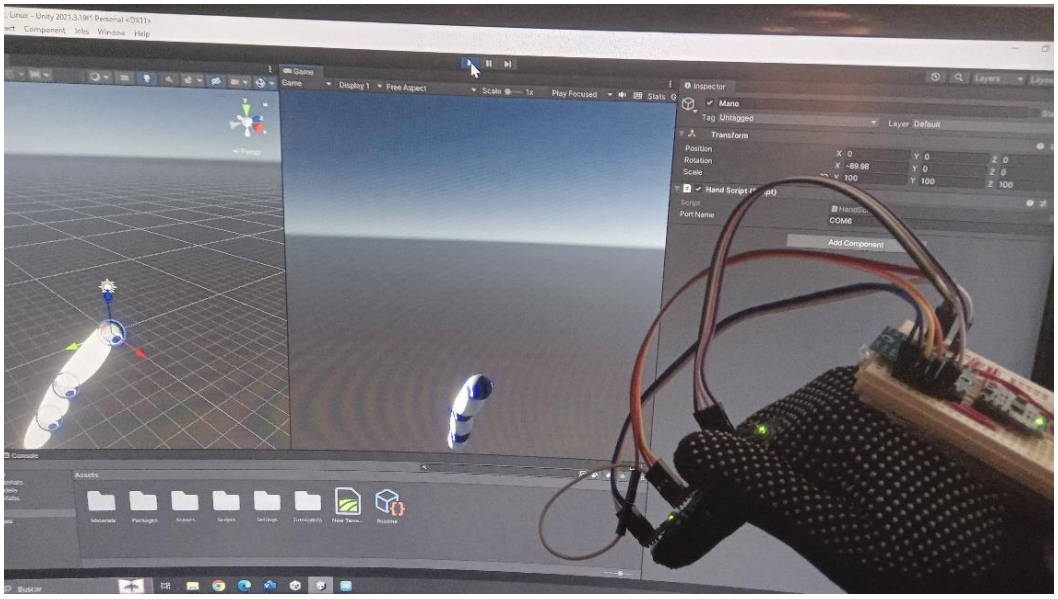


Figura 49: Evidencia 3 de las pruebas con el modelo funcional.

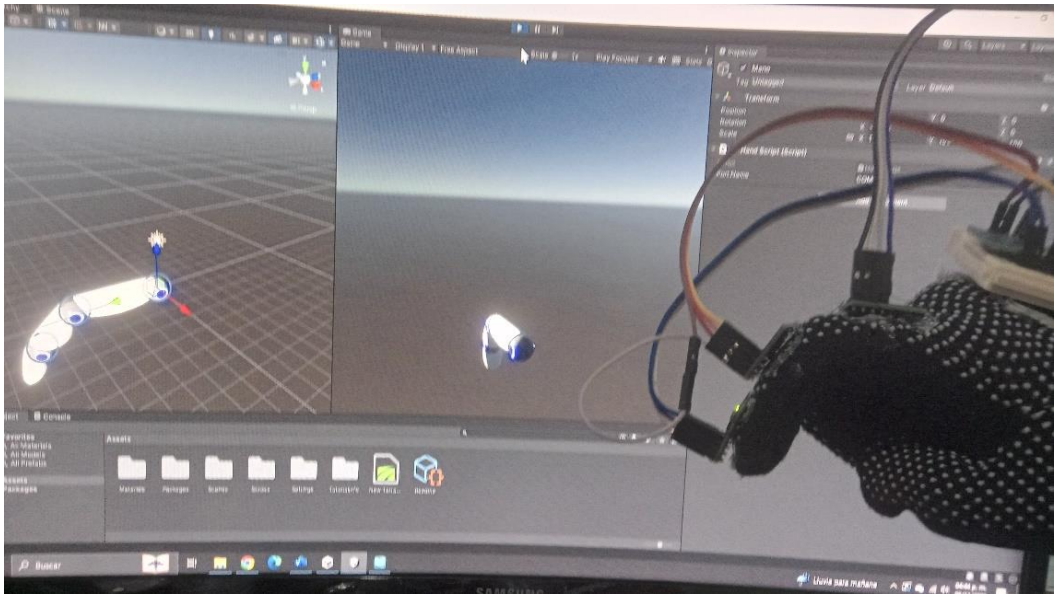


Figura 50: Evidencia 4 de las pruebas con el modelo funcional.

Con base en las evidencias fotográficas de la prueba, se observa que las posiciones de la mano son similares en cada imagen. Sin embargo, también se nota que, en cada foto, los sensores se desplazan de su posición original, lo que provoca pequeñas variaciones en los resultados esperados. Este problema se debe a la unión deficiente entre los componentes y los sensores, los cuales fueron adheridos al guante de tela con pegamento, lo que ocasiona que se despeguen con el movimiento de los dedos. Además, los cables tiraban de los sensores, contribuyendo a este inconveniente.

A pesar de estos desafíos, el modelo funcional arrojó resultados favorables, demostrando la eficacia del modelo cinemático propuesto y el diseño del guante. El siguiente paso sería avanzar hacia la fase de embodiment, donde se abordarían y resolverían estos problemas de unión y ubicación de los sensores para mejorar la precisión y confiabilidad del dispositivo.

CAPÍTULO 6: SISTEMA DE ADQUISICIÓN EN DETALLE

Una vez realizadas las pruebas, se logra ver que es posible mejorar el diseño con las tecnologías existentes, además de implementar otros subsistemas que ayudan a mejorar su rendimiento.

MICROCONTROLADOR Y FIRMWARE

El ESP32 demostró tener una capacidad notable para procesar datos. Además, este microcontrolador permite la adición de un sistema de aprendizaje y reconocimiento de patrones mediante la paquetería TensorFlow Lite de Python.

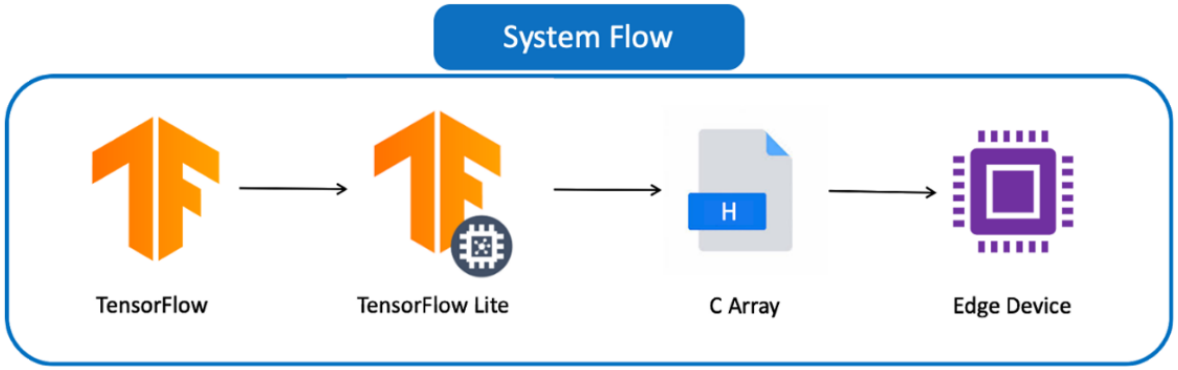


Figura 51: Diagrama de cómo se implementa TensorFlow en un microcontrolador [59].

Además, es posible mejorar la velocidad de transferencia al utilizar valores enteros. Por ejemplo, al considerar que el valor máximo del sensor 360 utiliza un total de 9 bits (sin considerar el bit de signo), los otros 7 bits del segundo byte son constantes en todo momento y se desperdician. Para optimizar esto, se puede "recortar" los bits innecesarios. En este proceso, primero se desplazan los bits que no son útiles y luego se les aplica un operador "*or*" en un arreglo de bytes inicializados en 0. Esto contribuye a enviar menos bits redundantes y mejorar la eficiencia de la transferencia de datos.

Conociendo los rangos de cada articulación, podemos determinar la cantidad máxima de bits necesaria, reduciendo aún más los bits. En casos donde los valores son negativos, podríamos agregarles una base para convertir el valor a un formato positivo. Este enfoque contribuiría a una mayor eficiencia en la transmisión de datos.

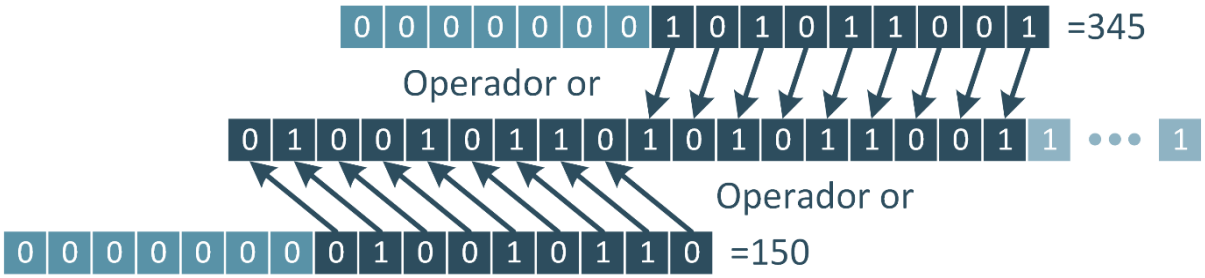


Figura 52: Nuevo sistema de transferencia de datos (bus de datos).

ALIMENTACIÓN POR PILAS

Una mejora para el guante podría ser la implementación de una alimentación mediante baterías LiPo, que son comunes en juguetes y drones. Se podría integrar un circuito capaz de cargar estas baterías y, a su vez, suministrar energía al guante. Un buen punto de referencia para este circuito sería basarse en el módulo TP4056, conocido por su capacidad para cargar baterías LiPo de manera eficiente y segura.

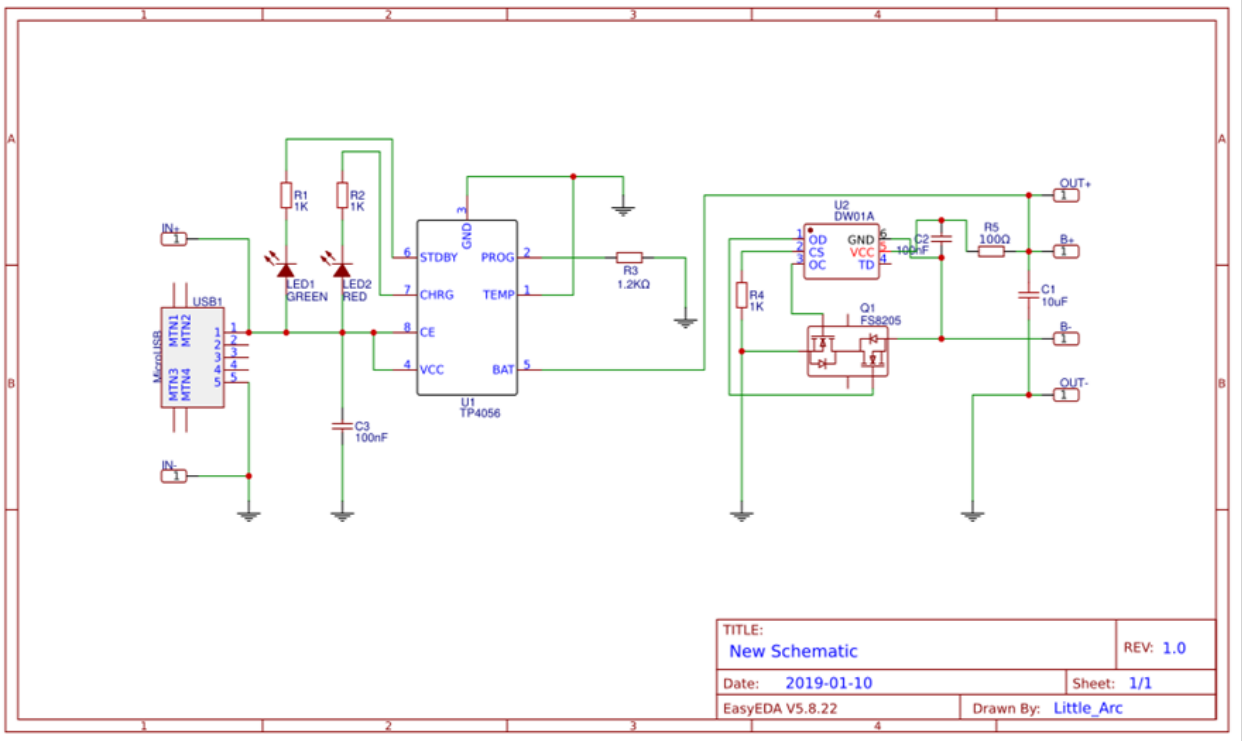


Figura 53: Diagrama de conexiones del TP4056 [2].

PLACA FLEXIBLE Y SENSORES

Debido a que las placas convencionales fabricadas con FR-4 son gruesas y rígidas, plantean problemas al implementarlas en el guante, ya que restringen la movilidad y añaden peso, como se evidenció en el modelo funcional. Para mejorar este diseño, se propone la impresión del circuito en una placa flexible. Las placas flexibles son delgadas, pueden deformarse y son altamente resistentes al calor. Este tipo de placas se aplicaría específicamente en las áreas que tienen impregnados los sensores IMU, ya que estos están dispersos por la mano.

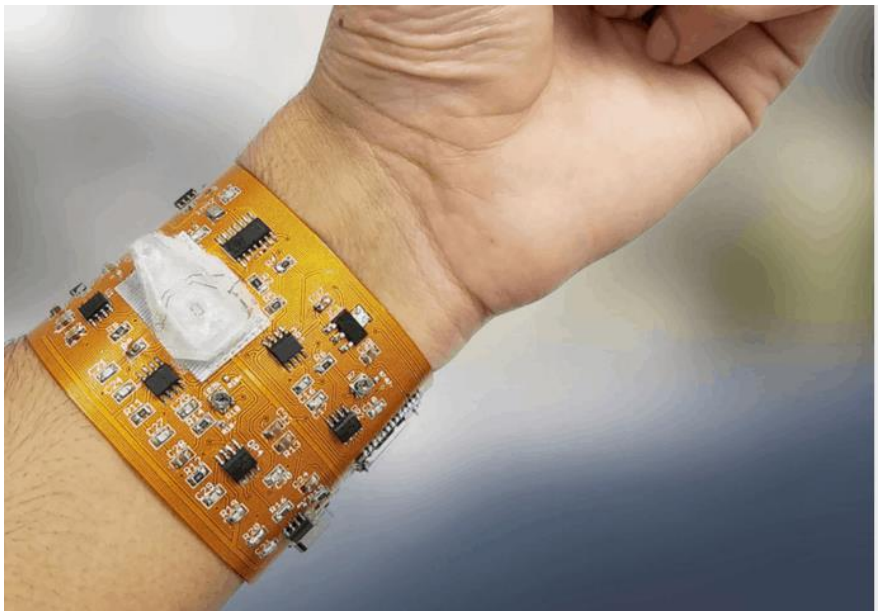


Figura 54: Ejemplo de una PCB flexible en una mano.

En el caso del MPU-6050, el sensor cuenta con un circuito simple en el que se integran solo capacitores, sin tomar en cuenta el regulador de voltaje incluido. Existen otros modelos de IMUs cuyas especificaciones pueden ser superiores al sensor mencionado anteriormente, aunque es importante tener en cuenta que sensores con mejores especificaciones también pueden ser más costosos.

Una desventaja del MPU-6050 es que solo cuenta con comunicación I2C y no SPI. En nuestro caso, sería más conveniente utilizar la comunicación SPI, ya que nos permite eliminar el multiplexor I2C, a pesar de utilizar más pines en el microcontrolador. Sin embargo, esto no representa un problema dado que el ESP32 tiene varios pines disponibles. Además, la comunicación SPI tiende a ser más rápida que la I2C, alcanzando frecuencias de hasta 25 MHz.

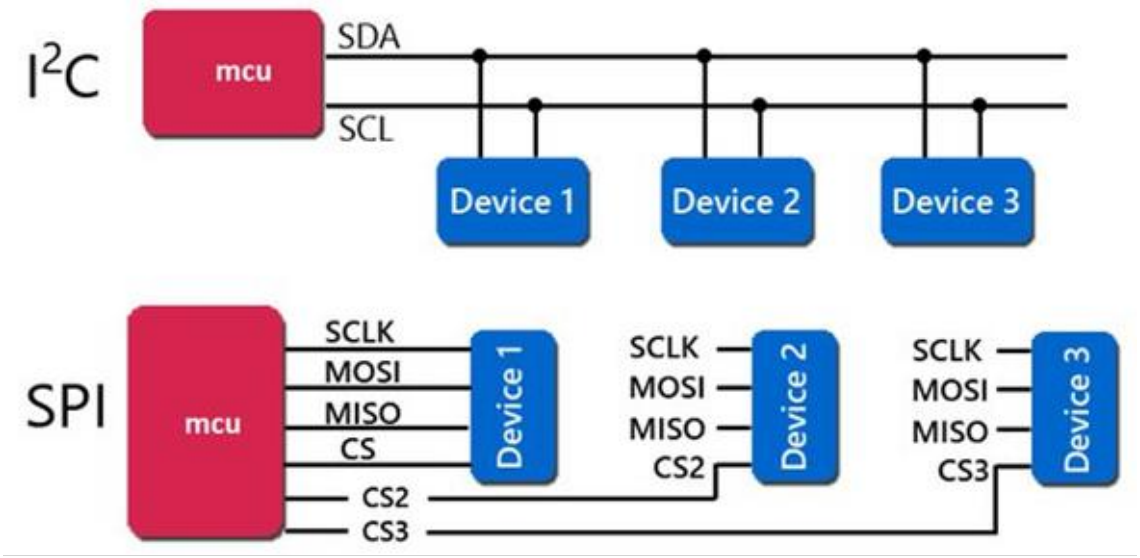


Figura 55: Comparación entre I2C y SPI [8].

CAPÍTULO 7: CONCLUSIONES Y TRABAJO FUTURO

Después de un análisis exhaustivo del proyecto, se puede concluir de manera satisfactoria que se logró el objetivo general establecido para este escrito. Se consiguió reducir significativamente el tiempo de procesamiento mediante la implementación de recursos económicos y de eficacia limitada. Cabe destacar el uso de un sensor que, aunque no es recomendable para nuevos diseños, demostró su utilidad en este contexto. Asimismo, un circuito improvisado, a pesar de sus defectos inherentes, mostró ser funcional y contribuyó de manera positiva a la optimización del proceso. En resumen, estos resultados respaldan la efectividad de la estrategia adoptada, destacando la importancia de considerar soluciones pragmáticas incluso cuando se utilizan recursos limitados.

Especificaciones técnicas

Especificación	Valor	Unidad
Resolucion angular	+/- 0.1	°
Rapidez angular	+/- 250	°/s
Distancia	5	m
Baud Rate	115200	baudios
Tasa de procesamiento	≈58.8	Hz
Datos de salida	23	-
Tasa de muestreo	60	Hz

Además, se logró confirmar la verosimilitud del modelo cinemático propuesto, presentando así una nueva propuesta para el diseño de robots que se inspira en los detalles de la mano. Aunque este diseño resultó exitoso, aún existen oportunidades para mejorarlo, especialmente al considerar más detalles específicos de cada dedo. La exploración de estos detalles requiere un estudio más profundo en el área médica, un campo que, aunque es un tanto ajeno a la ingeniería, puede aportar valiosa información para perfeccionar aún más el diseño. Este enfoque interdisciplinario sugiere un camino prometedor para futuras mejoras en la convergencia de la ingeniería y la medicina.

El objetivo principal de este proyecto no reside en la innovación o creación de un producto completamente nuevo. En cambio, se centra en aportar un valor agregado a los diseños existentes, específicamente mejorando las técnicas para obtener resultados más cercanos a la realidad. Este enfoque se traduce en una mayor exactitud en los resultados obtenidos, marcando así un nuevo paradigma en el proceso de diseño. Al no buscar la creación de algo completamente novedoso, sino más bien perfeccionar y enriquecer lo ya existente, el proyecto contribuye a elevar la calidad y precisión de las prácticas de diseño, proporcionando un enfoque renovado y más efectivo en la consecución de sus objetivos.

Adicionalmente, este proyecto abre la puerta a la mejora del diseño mediante la incorporación de inteligencia artificial. Al potenciar la transferencia de datos y centrarse especialmente en la Interfaz Hombre-Máquina (HMI), se propone un sistema que no solo modela la mano, sino que también identifica patrones de una lista de movimientos existentes, de manera análoga a la interfaz de una pantalla de un dispositivo inteligente. Esta ampliación de funcionalidades no solo incrementa la complejidad del diseño de manera positiva, sino que también sugiere un enfoque más avanzado y versátil para la aplicación de la tecnología, abriendo nuevas posibilidades para la interacción entre el usuario y el sistema.

Debido al diseño y la tecnología implementada, existe la posibilidad de obtener aún más información sobre el tema. Por ejemplo, la incorporación de un sensor de aceleración permite medir la fuerza aplicada por los dedos. Además, el sistema podría utilizarse para detectar ciertas condiciones médicas de la mano, como el síndrome del túnel carpiano, al identificar patrones específicos en los movimientos de las falanges. Este enfoque resultaría en un instrumento capaz de cuantificar de manera precisa las capacidades de la mano.

La conversión de este proyecto en un instrumento podría ser especialmente útil para verificar la ergonomía de dispositivos o maquinaria. Permitiría evaluar la comodidad de las posiciones de los dedos, así como analizar si las vibraciones de la maquinaria entran en resonancia con partes específicas del cuerpo humano. Este aspecto más amplio del proyecto no solo contribuiría al diseño eficiente y seguro de dispositivos, sino que también tendría aplicaciones significativas en la prevención y evaluación de posibles riesgos para la salud ocupacional.

La ingeniería mecatrónica no se limita a ser una especialidad, sino que destaca por ser una disciplina integral que abarca y fusiona varias áreas. Al enfrentarnos a problemas, nos capacita para comprender su origen y determinar la perspectiva que proporciona la respuesta a la problemática planteada.

REFERENCIAS

1. Agur, A. M., Daylei, A. F., & Moore, K. L. (2022). *Anatomía con orientación clínica*. Barcelona: Wolters Kluwer.
2. Arc, Little. (5 de Diciembre de 2023). *TP4056*. Obtenido de OSHW Lab: https://oshwlab.com/Little_Arc/TP4056
3. Avola, D., Cinque, L., Lacoviello, D., & Placidi, G. (2013). Overall design and implementation of the virtual glove. *Computers in Biology and Medicine*, 14.
4. Barrientos, A. (2007). *Fundamentos de robótica*. Madrid: Mc Graw Hill.
5. CaseGuard. (17 de Agosto de 2022). *¿A qué velocidad puede ver el ojo humano?* Obtenido de CaseGuard: https://caseguard.com/es/articles/cuantos-fotogramas-por-segundo-puede-ver-el-ojo-humano/
6. Cirque®. (2021). *Next Generation Input for VR Controllers*. Obtenido de Cirque®: https://www.cirque.com/cirque-vr-grip
7. Cosmar, M., Nickel, P., Schulz, R., & Zieschang, H. (27 de Octubre de 2022). *Human machine interface*. Obtenido de OSHWIKI: https://oshwiki.osha.europa.eu/es/themes/human-machine-interface
8. Draif, K. (21 de Junio de 2021). *Les protocoles de communication SPI, I2C et UART*. Obtenido de MaussaSoft: https://www.moussasoft.com/les-protocoles-de-communication-spi-i2c-et-uart
9. EDS Robotics. (20 de Julio de 2020). *Control de calidad por visión artificial: aplicaciones y ventajas*. Obtenido de EDS Robotics: https://www.edsrobotics.com/blog/control-calidad-vision-artificial/
10. Electronic Cats. (2023). *MPU6050*. Obtenido de GitHub: https://github.com/ElectronicCats/mpu6050
11. Fortuna y poder. (3 de Febrero de 2023). *Casas inteligentes ¿Qué son y cuántas hay en México?* Obtenido de Fortuna y poder:

https://fortunaypoder.com/entretenimiento/casa-inteligente-que-son-y-cuantas-hay-en-mexicoy

12. geekmomprojects. (2014 de Junio de 2014). *MPU-6050 Orientation from DMP vs. Complementary Filter*. Obtenido de Youtube: https://www.youtube.com/watch?v=2t-5CCyPJ74

13. Gutiérrez, A. L. (4 de Noviembre de 2021). Las empresas de electrónicos sufren por el desabasto en plena temporada alta. *Expansión*, pág. 1.

14. Hernández, L. d. (2017). *Visión artificial, OpenCV y Python*. Obtenido de Programa Facil: https://programarfacil.com/podcast/81-vision-artificial-opencv-phyton/

15. Hibbeler, R. C. (2010). *Ingeniería mecánica, Dinámica* (12a ed.). México: Pearson Prentice Hall.

16. Hibbeler, R. C. (2010). *Ingeniería mecánica, estática* (12a ed.). México: Pearson Prentice Hall.

17. HUAWEI. (2023). *¿Qué es una actualización OTA?* Obtenido de HUAWEI: https://consumer.huawei.com/es/support/faq/que-es-una-actualizacion-ota/

18. IMEPI. (2023). *Diseño de HMI: Maximizando la Seguridad y la Ergonomía en la Automatización Industrial*. Obtenido de IMEPI: https://imepi.com.mx/diseno-de-hmi-maximizando-la-seguridad-y-la-ergonomia-en-la-automatizacion-industrial/

19. IMEPI. (2023). *Explorando los Tipos de Pantallas HMI en la Automatización Industrial*. Obtenido de IMEPI: https://imepi.com.mx/explorando-los-tipos-de-pantallas-hmi-en-la-automatizacion-industrial/

20. Innovación y Tecnología. (3 de Octubre de 2022). *¿Qué es Alexa, google Home y Siri? Comparativa altavoces inteligentes amazon google y apple*. Obtenido de Innovación y Tecnología: https://innovacion-tecnologia.com/iot/que-es-alexa-google-home-y-siri-comparativa-y-diferencias/

21. IONOS Digital Guide . (13 de Enero de 2023). *FPS – Framerates en la TV, el cine y FPS en gaming*. Obtenido de IONOS Digital Guide : https://www.ionos.mx/digitalguide/servidores/know-how/fps/

22. Kapandji, A. I. (2006). *Fisiología Articular*. Madrid: Editorial Médica Panamericana.

23. KenHub. (2023). *Huesos de la muñeca y de la mano*. Obtenido de KenHub: https://www.kenhub.com/es/study/huesos-de-la-muneca-y-de-la-mano

24. KenHub. (2023). *Ligamentos de la muñeca y de la mano*. Obtenido de KenHub: https://www.kenhub.com/es/study/ligamentos-de-la-muneca-y-de-la-mano

25. KenHub. (2023). *Mano y muñeca (anatomía)*. Obtenido de KenHub: https://www.kenhub.com/es/library/anatomia-es/mano-y-muneca

26. KenHub. (2023). *Músculos de la mano*. Obtenido de KenHub: https://www.kenhub.com/es/study/musculos-de-la-mano

27. KenHub. (2023). *Tipos de movimientos del cuerpo humano*. Obtenido de KenHub: https://www.kenhub.com/es/library/anatomia-es/tipos-de-movimientos-del-cuerpo-humano

28. Lecturio. (18 de Abril de 2023). *Hand: Anatomy*. Obtenido de Lecturio: https://app.lecturio.com/#/article/3594

29. Lecturio. (29 de Diciembre de 2023). *Wrist Joint: Anatomy*. Obtenido de Lecturio: https://app.lecturio.com/#/article/2866

30. Little_Arc. (12 de Diciembre de 2023). *OSHWLab*. Obtenido de TP4056: https://oshwlab.com/Little_Arc/TP4056

31. MANUS™. (2023). *QUANTUM MOCAP METAGLOVES®*. Obtenido de MANUS™: https://www.manus-meta.com/products/quantum-mocap-metagloves

32. Miller Pérez, C., Sánchez Islas, E., Mucio Ramírez, S., Mendoza Sotelo, J., & León Olea, M. (23 de Febrero de 2009). Los contaminantes ambientales bifenilos policlorinados (PCB) y sus efectos sobre el Sistema Nervioso y la salud. *Salud Mental*, 12. Obtenido de https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0185-33252009000400009

33. Random Nerd Tutorials. (2023). *Guide for TCA9548A I2C Multiplexer: ESP32, ESP8266, Arduino*. Obtenido de Random Nerd Tutorials: https://randomnerdtutorials.com/tca9548a-i2c-multiplexer-esp32-esp8266-arduino/

34. Rivero, D. T. (8 de Julio de 2022). *La obsolescencia tecnológica: un breve resumen*. Obtenido de Do Better: https://dobetter.esade.edu/es/la-obsolescencia-tecnologica-un-breve-resumen

35. Robotic-Lab. (2023). *Guante que traduce el lenguaje de signos*. Obtenido de Robotic-Lab: https://www.robotic-lab.com/blog/2012/07/11/guante-que-traduce-el-lenguaje-de-signos/

36. Sanket, N. (2023). *Madgwick Filter*. Obtenido de Nitin J. Sanket: https://nitinjsanket.github.io/tutorials/attitudeest/madgwick.html

37. Sanket, N. (2023). *Mahony Filter*. Obtenido de Nitin J. Sanket: https://nitinjsanket.github.io/tutorials/attitudeest/mahony.html

38. Shiva Shankar.s, M. G. (2 de Junio de 2020). *An Introduction to Edge Computing On ESP32*. Obtenido de Hackster.io: https://www.hackster.io/edge-c/an-introduction-to-edge-computing-on-esp32-6f3834

39. Silva Rico, J. A. (2022). Dinámica de Maquinaria. *(Notas de clase)*. Facultad de Ingeniería, UNAM, Ciudad de México.

40. Somma, F. J. (s.f.). Cuaterniones y ángulos de Euler para describir rotaciones en R3. *(Tesis de Licenciatura)*. Universidad Abierta Interamericana, Rosario.

41. Stanford Medicine Children's Health. (2023). *Anatomía de la mano*. Obtenido de Stanford Medicine Children's Health: https://www.stanfordchildrens.org/es/topic/default?id=anatomyofthehand-85-P04195

42. TDK InvenSense. (2023). *MPU-6000 and MPU-6050 Product Specification Revision 3.4*. Obtenido de TDK InvenSense: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

43. TDK InvenSense. (2023). *MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Device*. Obtenido de TDK InvenSense: https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/

44. Telleen, S. (2022). *Human Anatomy*. OpenStax CNX.

45. Wikipedia. (30 de Septiembre de 2023). *Captura de movimiento*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Captura_de_movimiento

46. Wikipedia. (23 de Noviembre de 2023). *Cuaternión*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Cuaterni%C3%B3n

47. Wikipedia. (15 de Mayo de 2023). *Kinesiología*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Kinesiolog

48. Wikipedia. (21 de Noviembre de 2023). *Mano*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Mano

ANEXOS

CÓDIGO DEL MODELO FUNCIONAL

```
// Importamos las bibliotecas correspondientes
#include <WiFi.h>
#include <Ticker.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include "BluetoothSerial.h"
#include "MPU6050_6Axis.h"
#include "Config.h"

// Definimos para cambiar la comunicación a USB o Bluetooth
// #define BLUETOOTH

// Creamos los subprocesos (hilos)
Ticker Process;     // Proceso principal
Ticker Initialize;  // Inicializamos en núcleo 1

// Variable de verificación de actualización
bool update = true;

// Variable para comunicación SerialBT
BluetoothSerial SerialBT;

// Declaración de objetos correspondientes a los sensores
MPU6050 palm;
MPU6050 proximalPhalange;
MPU6050 intermediatePhalange;
MPU6050 distalPhalange;

// Declaramos constantes de ángulos oblicuos
const int8_t fi1 = 15;  // En grados
const int8_t fi2 = 10;  // En grados
const int8_t fi3 = 10;  // En grados

// Declaración de objetos cuaterniones
Quaternion q1,q2,q3,q4; // [w, x, y, z]

// Declaramos el buffer de datos para los MPU-6050
uint8_t fifoBuffer[64]; // FIFO de datos

// Función para transferir datos de forma rápida
void fastSerial(int16_t *in) {
  // Declaramos dos casos posibles para alternar entre USB y Bluetooth
  // Convertimos de int16_t a uint8_t (bytes)
  #ifdef BLUETOOTH
    // Transferencia Bluetooth
    SerialBT.write(208); // Valor de referencia
    for (uint8_t i = 0; i < 7; i++) {
      SerialBT.write((byte) (in[i] >> 8));
      SerialBT.write((byte) (in[i]));
    }
  #else
  // Transferencia USB
    Serial.write(208);  // Valor de referencia
```

```
    for (uint8_t i = 0; i < 7; i++) {
      Serial.write((byte) (in[i] >> 8));
      Serial.write((byte) (in[i]));
    }
  #endif
}

// Función para inicializar los MPU-6050
void initializeMPU6050(MPU6050 *sensor) {
  sensor->initialize();                                          // Inicializamos el sensor
  sensor->setFullScaleAccelRange(0);                             // Escala +-2g
  if (sensor->testConnection()) Serial.println("Connection Success!");  // conexión exitosa
  else Serial.println("Connection Failed!");                     // conexión fallida
  sensor->dmpInitialize();                                       // Inicializamos el DMP
  sensor->CalibrateAccel(6);                                     // Calibramos el acelerómetro
  sensor->CalibrateGyro(6);                                      // Calibramos el giroscopio
  sensor->setDMPEnabled(true);                                   // Habilitamos el DMP
  Serial.println();
}

// Función para habilitar carga el firmware vía OTA
void InitializeOTA() {
  // Seguridad para cargar el firmware vía OTA
  ArduinoOTA.setPort(3232);             // Puerto por defecto: 3232
  ArduinoOTA.setHostname("MyESP32");    // Declaramos un Hostname
  ArduinoOTA.setPassword(OTApassword);  // Contraseña para cargar

  // Código por defecto
  ArduinoOTA
    .onStart([]() {
      String type;
      if (ArduinoOTA.getCommand() == U_FLASH)
        type = "sketch";
      else // U_SPIFFS
        type = "filesystem";
      // NOTA: Si actualiza SPIFFS, este sería el lugar para desmontar SPIFFS usando SPIFFS.end()
      Serial.println("Start updating " + type);
    })
    .onEnd([]() {
      Serial.println("\nEnd");
    })
    .onProgress([](unsigned int progress, unsigned int total) {
      Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
    })
    .onError([](ota_error_t error) {
      Serial.printf("Error[%u]: ", error);
      if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
      else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
      else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
      else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
      else if (error == OTA_END_ERROR) Serial.println("End Failed");
    });

  // Inicializamos OTA
  ArduinoOTA.begin();

  // Obtenemos la IP local
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}
```

```
// Función para inicializar conexión wifi
void ConnectWiFi_STA() {
  // Iniciamos conexión wifi
  Serial.println("Booting");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  // Verificamos conexión con modem
  if (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Inactive update");
    update = false;    // No habilitados carga vía OTA
  } else {
    Serial.println("Connection Success! Update available");
    InitializeOTA();  // Habilitados carga vía OTA
  }
}

// Proceso para extraer datos de los sensores
void Glove(){
  // Bus de datos en ceros
  int16_t data[] = {0,0,0,0,0,0,0};

  // Extraemos datos del DMP del primer MPU-6050
  TCA9548A(7);
  palm.dmpGetCurrentFIFOPacket(fifoBuffer);
  palm.dmpGetQuaternion(&q1, fifoBuffer);

  // Extraemos datos del DMP del segundo MPU-6050
  TCA9548A(6);
  proximalPhalange.dmpGetCurrentFIFOPacket(fifoBuffer);
  proximalPhalange.dmpGetQuaternion(&q2, fifoBuffer);

  // Extraemos datos del DMP del tercero MPU-6050
  TCA9548A(5);
  intermediatePhalange.dmpGetCurrentFIFOPacket(fifoBuffer);
  intermediatePhalange.dmpGetQuaternion(&q3, fifoBuffer);

  // Extraemos datos del DMP del cuarto MPU-6050
  TCA9548A(4);
  distalPhalange.dmpGetCurrentFIFOPacket(fifoBuffer);
  distalPhalange.dmpGetQuaternion(&q4, fifoBuffer);

  // Cambiamos el sistema de referencia de inercial a local
  q4 = q3.getConjugate().getProduct(q4);  // F. Medial a F. Distal
  q3 = q2.getConjugate().getProduct(q3);  // F. Medial a F. Distal
  q2 = q1.getConjugate().getProduct(q2);  // Palmar a F. Proximal

  // Calculamos el valor de los ángulos
  // NOTA: Se extrajo de un despeje de ecuaciones planteadas con anterioridad
  data[0] = round(atan2(2*(q1.x*q1.y - q1.w*q1.z),2*(q1.w*q1.w + q1.x*q1.x)-1)*1800/M_PI);     //α
  data[1] = -round(asin(2*(q1.x*q1.z + q1.w*q1.y))*1800/M_PI);                                 //β
  data[2] = round(atan2(2*(q1.y*q1.z - q1.w*q1.x),2*(q1.w*q1.w + q1.z*q1.z)-1)*1800/M_PI);     //γ
  data[3] = -round(2*atan2(q2.x,q2.z*sin(fi1*M_PI/180) + q2.w*cos(fi1*M_PI/180))*1800/M_PI);   //θ1
  data[4] = round(2*atan2(q2.z,q2.w) * 1800/M_PI);                                            //ψ
  data[5] = -round(2*atan2(q3.x,q3.w * cos(fi2*M_PI/180))* 1800/M_PI);                        //θ2
  data[6] = -round(2*atan2(q4.x,q4.w * cos(fi3*M_PI/180))* 1800/M_PI);                        //θ3

  // Mandamos el arreglo de forma rápida
  fastSerial(data);
```

```
}

// Uso de TCA9548A
void TCA9548A(uint8_t bus){
  Wire.beginTransmission(0x70); // TCA9548A dirección es 0x70
  Wire.write(1 << bus);         // Buscamos el byte selector
  Wire.endTransmission();       // Cerramos transmisión
}

// Función para inicializar núcleo 1
void Begin() {
  Wire.begin(18, 19, 400000); // SDA, SCL
  SerialBT.begin("MyESP32");  // Bluetooth MyESP32
  TCA9548A(7);                // Selector en 7
  initializeMPU6050(&palm);
  TCA9548A(6);                // Selector en 6
  initializeMPU6050(&proximalPhalange);
  TCA9548A(5);                // Selector en 5
  initializeMPU6050(&intermediatePhalange);
  TCA9548A(4);                // Selector en 4
  initializeMPU6050(&distalPhalange);
}

// Configuramos el ESP32
void setup() {
  Serial.begin(115200);         // Serial 115200 baudios
  ConnectWiFi_STA();            // Iniciamos conexión wifi y OTA
  Initialize.once_ms(0,Begin);  // Inicializamos ESP32
  Process.attach_ms(17, Glove); // Declaramos el hilo a 17 ms (≈60 Hz)
}

// Loop para cargar código vía OTA (núcleo 0)
void loop() {
  if (update) ArduinoOTA.handle();
}
```

CÓDIGO DE PRUEBAS DE PROCESAMIENTO

```
// Importamos las bibliotecas correspondientes
#include "MPU6050_6Axis.h"

// Cronómetros
unsigned long time1 = 0;

// Declaración de objetos correspondientes a los sensores
MPU6050 palm;
MPU6050 proximalPhalange;
MPU6050 intermediatePhalange;
MPU6050 distalPhalange;

// Declaramos constantes de ángulos oblicuos
const int8_t fi1 = 15;  // En grados
const int8_t fi2 = 10;  // En grados
const int8_t fi3 = 10;  // En grados

// Declaración de objetos cuaterniones
Quaternion q1,q2,q3,q4; // [w, x, y, z]

// Declaramos el buffer de datos para los MPU-6050
uint8_t fifoBuffer[64]; // FIFO de datos

// Función para transferir datos de forma rápida
void fastSerial(int16_t *in) {
  // Transferencia USB
  Serial.write(208); // Valor de referencia
  for (uint8_t i = 0; i < 7; i++) {
    Serial.write((byte) (in[i] >> 8));
    Serial.write((byte) (in[i]));
  }
}

// Función para inicializar los MPU-6050
void initializeMPU6050(MPU6050 *sensor) {
  sensor->initialize();                                          // Inicializamos el sensor
  sensor->setFullScaleAccelRange(0);                             // Escala +-2g
  if (sensor->testConnection()) Serial.println("Connection Success!");  // conexión exitosa
  else Serial.println("Connection Failed!");                     // conexión fallida
  sensor->dmpInitialize();                                       // Inicializamos el DMP
  sensor->CalibrateAccel(6);                                     // Calibramos el acelerómetro
  sensor->CalibrateGyro(6);                                      // Calibramos el giroscopio
  sensor->setDMPEnabled(true);                                   // Habilitamos el DMP
  Serial.println();
}

// Uso de TCA9548A
void TCA9548A(uint8_t bus){
  Wire.beginTransmission(0x70); // TCA9548A dirección es 0x70
  Wire.write(1 << bus);         // Buscamos el byte selector
  Wire.endTransmission();       // Cerramos transmisión
}

// Función para inicializar núcleo 1
void Begin() {
  Wire.begin(18, 19, 400000); // SDA, SCL
  TCA9548A(7);                // Selector en 7
  initializeMPU6050(&palm);
```

```
  TCA9548A(6);                // Selector en 6
  initializeMPU6050(&proximalPhalange);
  TCA9548A(5);                // Selector en 5
  initializeMPU6050(&intermediatePhalange);
  TCA9548A(4);                // Selector en 4
  initializeMPU6050(&distalPhalange);
}

// Proceso nuevo
void newProcess(){
  // Bus de datos en ceros
  int16_t data[] = {0,0,0,0,0,0,0};

  // Extraemos datos del DMP del primer MPU
  TCA9548A(7);
  palm.dmpGetCurrentFIFOPacket(fifoBuffer);
  palm.dmpGetQuaternion(&q1, fifoBuffer);

  // Extraemos datos del DMP del segundo MPU
  TCA9548A(6);
  proximalPhalange.dmpGetCurrentFIFOPacket(fifoBuffer);
  proximalPhalange.dmpGetQuaternion(&q2, fifoBuffer);

  // Extraemos datos del DMP del tercero MPU
  TCA9548A(5);
  intermediatePhalange.dmpGetCurrentFIFOPacket(fifoBuffer);
  intermediatePhalange.dmpGetQuaternion(&q3, fifoBuffer);

  // Extraemos datos del DMP del cuarto MPU
  TCA9548A(4);
  distalPhalange.dmpGetCurrentFIFOPacket(fifoBuffer);
  distalPhalange.dmpGetQuaternion(&q4, fifoBuffer);

  // Cambiamos el sistema de referencia de inercial a local
  q4 = q3.getConjugate().getProduct(q4);  // F. Medial a F. Distal
  q3 = q2.getConjugate().getProduct(q3);  // F. Medial a F. Distal
  q2 = q1.getConjugate().getProduct(q2);  // Palmar a F. Proximal

  // Calculamos el valor de los ángulos
  // NOTA: Se extrajo de un despeje de ecuaciones planteadas con anterioridad
  data[0] = round(atan2(2*(q1.x*q1.y - q1.w*q1.z),2*(q1.w*q1.w + q1.x*q1.x)-1)*1800/M_PI);    //α
  data[1] = -round(asin(2*(q1.x*q1.z + q1.w*q1.y))*1800/M_PI);                                //β
  data[2] = round(atan2(2*(q1.y*q1.z - q1.w*q1.x),2*(q1.w*q1.w + q1.z*q1.z)-1)*1800/M_PI);    //γ
  data[3] = -round(2*atan2(q2.x,q2.z*sin(fi1*M_PI/180) + q2.w*cos(fi1*M_PI/180))*1800/M_PI);  //θ1
  data[4] = round(2*atan2(q2.z,q2.w) * 1800/M_PI);                                           //ψ
  data[5] = -round(2*atan2(q3.x,q3.w * cos(fi2*M_PI/180))* 1800/M_PI);                       //θ2
  data[6] = -round(2*atan2(q4.x,q4.w * cos(fi3*M_PI/180))* 1800/M_PI);                       //θ3

  // Mandamos el arreglo de forma rápida
  fastSerial(data);
}

// Proceso viejo
void oldProcess(){
  // Extraemos datos del DMP del primer MPU
  TCA9548A(7);
  palm.dmpGetCurrentFIFOPacket(fifoBuffer);
  palm.dmpGetQuaternion(&q1, fifoBuffer);

  // Extraemos datos del DMP del segundo MPU
```

```
  TCA9548A(6);
  proximalPhalange.dmpGetCurrentFIFOPacket(fifoBuffer);
  proximalPhalange.dmpGetQuaternion(&q2, fifoBuffer);

  // Extraemos datos del DMP del tercero MPU
  TCA9548A(5);
  intermediatePhalange.dmpGetCurrentFIFOPacket(fifoBuffer);
  intermediatePhalange.dmpGetQuaternion(&q3, fifoBuffer);

  // Extraemos datos del DMP del cuarto MPU
  TCA9548A(4);
  distalPhalange.dmpGetCurrentFIFOPacket(fifoBuffer);
  distalPhalange.dmpGetQuaternion(&q4, fifoBuffer);

  // Cambiamos el sistema de referencia de inercial a local
  q4 = q3.getConjugate().getProduct(q4);  // F. Medial a F. Distal
  q3 = q2.getConjugate().getProduct(q3);  // F. Medial a F. Distal
  q2 = q1.getConjugate().getProduct(q2);  // Palmar a F. Proximal

  Serial.print(String(208) + ",");  // Valor de referencia
  printQuaternion(q1);              // IMU 1
  printQuaternion(q2);              // IMU 2
  printQuaternion(q3);              // IMU 3
  printQuaternion(q4);              // IMU 4
}

// Función para imprimir cuaterniones
void printQuaternion(Quaternion q){
  Serial.print(String(q.w) + ",");  //[w]
  Serial.print(String(q.x) + ",");  //[x]
  Serial.print(String(q.y) + ",");  //[y]
  Serial.print(String(q.z) + ",");  //[z]
}

// Configuramos el ESP32
void setup() {
  Serial.begin(115200);   // Serial 115200 baudios
  Begin();                // Subrutina para inicializar

  // Inicializamos el contador en 0, y ejecutamos el algoritmo nuevo
  int count = 0;
  time1 = millis();
  do{
    count++;
    newProcess();
  } while(millis()-time1 < 1000);
  Serial.println("\nIteraciones por segundo algoritmo nuevo: " + String(count));

  // Inicializamos el contador en 0, y ejecutamos el algoritmo viejo
  count = 0;
  time1 = millis();
  do{
    count++;
    oldProcess();
  } while(millis()-time1 < 1000);
  Serial.println("\nIteraciones por segundo algoritmo viejo: " + String(count));

}

// Loop vacío (núcleo 0)
```

```
void loop() {
  delay(16); // El loop no se usa
}
```

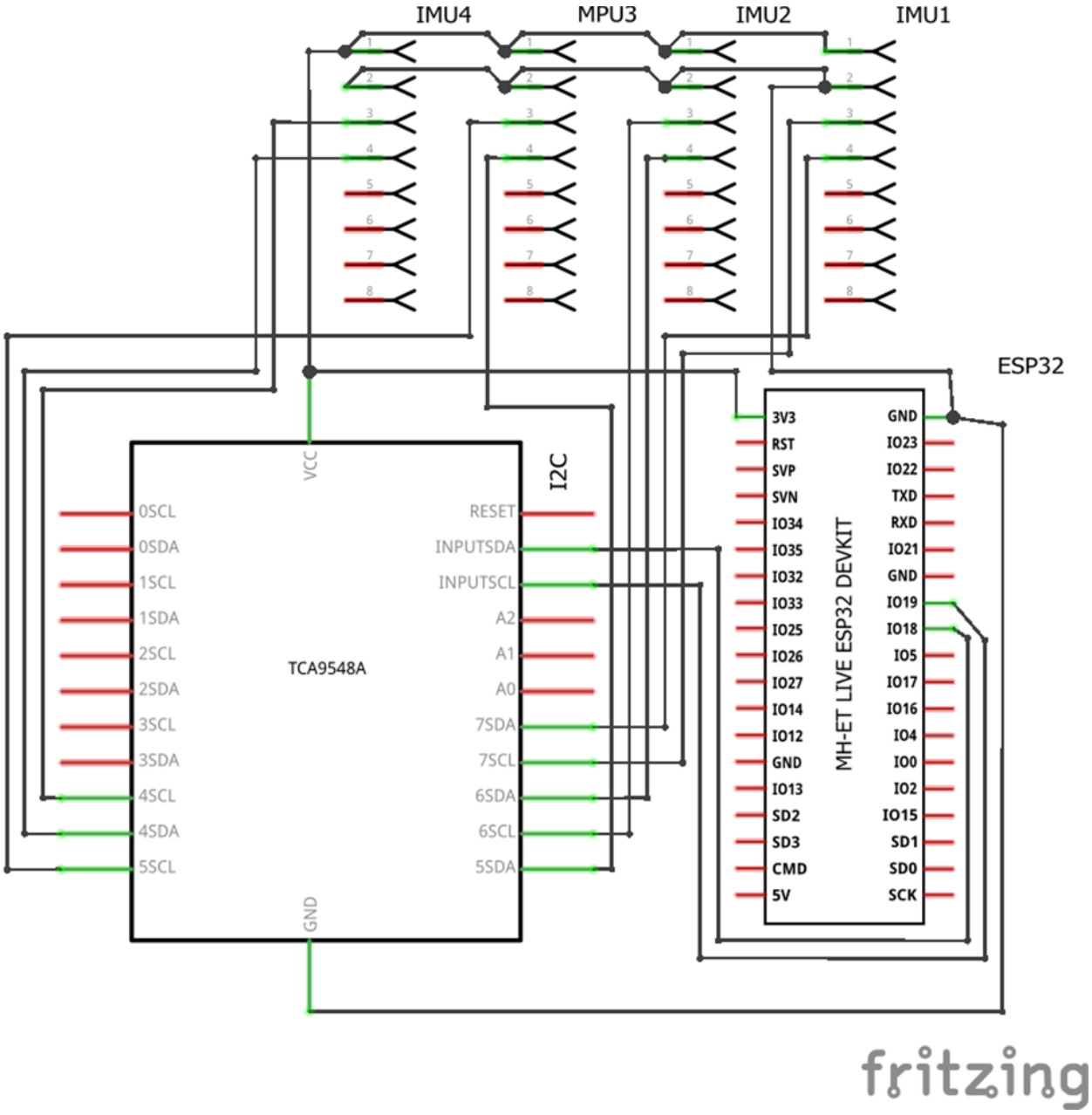
CÓDIGO DE UNITY

```
// Importamos las bibliotecas correspondientes
using System;
using System.IO.Ports;
using System.Threading;
using UnityEngine;

public class HandScript : MonoBehaviour
{
    // Nombre del puerto serial
    public string portName;

    // Objetos relacionados con la comunicacion serial
    private bool abort;
    private SerialPort serialPort;
    private Thread serialThread;

    // Ejes de rotación oblicuos
    private Vector3 vectorFi1;
    private Vector3 vectorFi2;
    private Vector3 vectorFi3;

    // Constantes de ángulos oblicuos
    private const float fi1 = 15;
    private const float fi2 = 10;
    private const float fi3 = 10;

    // Ángulos de flexión y extensión
    private float th1, psi, th2, th3;


    // Función de inicio
    void Start()
    {
        // Asignamos los ejes oblicuos de rotación
        vectorFi1 = Quaternion.AngleAxis(fi1, Vector3.up) * Vector3.left;
        vectorFi2 = Quaternion.AngleAxis(fi2, Vector3.up) * Vector3.left;
        vectorFi3 = Quaternion.AngleAxis(fi3, Vector3.up) * Vector3.left;

        // Inicializamos el hilo y la comunicación serial
        serialThread = new Thread(Read);
        serialPort = new SerialPort(portName, 115200);

        try
        {
            // Abrimos comunicación serial
            serialPort.Open();
            serialPort.DiscardOutBuffer();
            serialPort.DiscardInBuffer();
            serialThread.Start();
        }
        catch (Exception)
```

```
    {
        // Mandamos mensaje en caso de no detectar dispositivo
        Debug.Log("No se detecta el dispositivo");
    }
}


// Actualizamos las transformaciones cada frame
void Update()
{
    // Falange proximal
    transform.GetChild(0).localRotation =
        Quaternion.AngleAxis(th1, vectorFi1) *
        Quaternion.AngleAxis(-psi, Vector3.forward);

    // Falange medial
    transform.GetChild(0).GetChild(0).localRotation =
        Quaternion.AngleAxis(th2, vectorFi2);

    // Falange distal
    transform.GetChild(0).GetChild(0).GetChild(0).localRotation =
        Quaternion.AngleAxis(th3, vectorFi3);

}

// Subrutina para leer los datos seriales
void Read()
{
    // Inicializamos bus de datos
    byte[] bus = new byte[14];
    while (true)
    {
        // Cerramos la comunicación serial y abortamos la subrutina
        if (abort)
        {
            serialPort.DiscardOutBuffer();
            serialPort.DiscardInBuffer();
            serialPort.Close();
            serialThread.Abort();
            break;
        }

        // Lectura de datos y verificación de conexión
        try
        {
            if (serialPort.BytesToRead >= 15)
            {
                // Verificamos el valor de referencia
                serialPort.Read(bus, 0, 1);
                if (bus[0] != 208) continue;

                // Leemos el bus de datos y convertimos
                serialPort.Read(bus, 0, 14);
```

```
                Array.Reverse(bus);
                th1 = BitConverter.ToInt16(bus, 6) / 10.0f;
                psi = BitConverter.ToInt16(bus, 4) / 10.0f;
                th2 = BitConverter.ToInt16(bus, 2) / 10.0f;
                th3 = BitConverter.ToInt16(bus, 0) / 10.0f;
            }
        }
        catch (Exception)
        {
            Debug.Log("No se detecta el dispositivo");
        }
    }
}

// Abortamos el programa
private void OnApplicationQuit()
{
    abort = true;
}
}
```

ESQUEMÁTICO ELECTRÓNICO DEL MODELO FUNCIONAL



Modelo Cinemático de la Mano

```
In[ ]:= $PrePrint = # /. {Csc[z_] :> 1 / Defer@Sin[z], Sec[z_] :> 1 / Defer@Cos[z]} &;
        (*Formato de sec a cos*)
```

Cuaterniones

```
In[ ]:= << Quaternions`

In[ ]:= ToVector[q_] := Table[q[[i]], {i, 1, 4}]

        RotationQuaternion[θ_, v_] :=
         Quaternion[Cos[θ / 2],
          Sin[θ / 2] Normalize[v][[1]],
          Sin[θ / 2] Normalize[v][[2]],
          Sin[θ / 2] Normalize[v][[3]]]
```

Matrices de rotación

```
In[ ]:= M1 = EulerMatrix[{ϕ1, -θ1, ψ - ϕ1}, {3, 1, 3}] // Simplify;
        M2 = EulerMatrix[{ϕ2, -θ2, -ϕ2}, {3, 1, 3}] // Simplify;
        M3 = EulerMatrix[{ϕ3, -θ3, -ϕ3}, {3, 1, 3}] // Simplify;
```

```
In[•]:= Format[M1, TraditionalForm]
        formato   forma tradicional
        Format[M2, TraditionalForm]
        formato   forma tradicional
        Format[M3, TraditionalForm]
        formato   forma tradicional
```

Out[•]=

$$\begin{pmatrix} c(\phi_1)\,c(\psi-\phi_1)-c(\Theta_1)\,s(\phi_1)\,s(\psi-\phi_1) & -c(\Theta_1)\,s(\phi_1)\,c(\psi-\phi_1)-c(\phi_1)\,s(\psi-\phi_1) & -s(\Theta_1)\,s(\phi_1) \\ c(\Theta_1)\,c(\phi_1)\,s(\psi-\phi_1)+s(\phi_1)\,c(\psi-\phi_1) & c(\Theta_1)\,c(\phi_1)\,c(\psi-\phi_1)-s(\phi_1)\,s(\psi-\phi_1) & c(\phi_1)\,s(\Theta_1) \\ -s(\Theta_1)\,s(\psi-\phi_1) & s(\Theta_1)\,(-c(\psi-\phi_1)) & c(\Theta_1) \end{pmatrix}$$

Out[•]=

$$\begin{pmatrix} c(\Theta_2)\sin^2(\phi_2)+\cos^2(\phi_2) & \sin^2\left(\frac{\Theta_2}{2}\right)s(2\phi_2) & -s(\Theta_2)\,s(\phi_2) \\ \sin^2\left(\frac{\Theta_2}{2}\right)s(2\phi_2) & c(\Theta_2)\cos^2(\phi_2)+\sin^2(\phi_2) & c(\phi_2)\,s(\Theta_2) \\ s(\Theta_2)\,s(\phi_2) & -c(\phi_2)\,s(\Theta_2) & c(\Theta_2) \end{pmatrix}$$

Out[•]=

$$\begin{pmatrix} c(\Theta_3)\sin^2(\phi_3)+\cos^2(\phi_3) & \sin^2\left(\frac{\Theta_3}{2}\right)s(2\phi_3) & -s(\Theta_3)\,s(\phi_3) \\ \sin^2\left(\frac{\Theta_3}{2}\right)s(2\phi_3) & c(\Theta_3)\cos^2(\phi_3)+\sin^2(\phi_3) & c(\phi_3)\,s(\Theta_3) \\ s(\Theta_3)\,s(\phi_3) & -c(\phi_3)\,s(\Theta_3) & c(\Theta_3) \end{pmatrix}$$

Ángulos de la mano (Matrices de rotación)

```
In[•]:= M = {{2 q0^2 + 2 q1^2 - 1, 2 (q1 q2 - q0 q3), 2 (q0 q2 + q1 q3)},
             {2 (q1 q2 + q0 q3), 2 q0^2 + 2 q2^2 - 1, 2 (q2 q3 - q0 q1)},
             {2 (q1 q3 - q0 q2), 2 (q0 q1 + q2 q3), 2 q0^2 + 2 q3^2 - 1}};

In[•]:= Format[M, TraditionalForm]
        formato   forma tradicional
```

Out[•]=

$$\begin{pmatrix} 2q_0^2+2q_1^2-1 & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_1q_2+q_0q_3) & 2q_0^2+2q_2^2-1 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & 2q_0^2+2q_3^2-1 \end{pmatrix}$$

```
In[•]:=
        ψm = Solve[M1[[3, 1]] / M1[[3, 2]] == M[[3, 1]] / M[[3, 2]], ψ][[1, 1]] /. c1 → 0;
             resuelve
        Θm1 = Solve[M1[[2, 3]] / M1[[3, 3]] == M[[2, 3]] / M[[3, 3]], Θ1][[1, 1]] /. c1 → 0;
              resuelve
        Θm2 = Solve[M2[[2, 3]] / M2[[3, 3]] == M[[2, 3]] / M[[3, 3]], Θ2][[1, 1]] /. c1 → 0;
              resuelve
        Θm3 = Solve[M3[[2, 3]] / M3[[3, 3]] == M[[2, 3]] / M[[3, 3]], Θ3][[1, 1]] /. c1 → 0;
              resuelve
```

```
In[•]:= Format[ψm, TraditionalForm]
        formato   forma tradicional
        Format[θm1, TraditionalForm]
        formato   forma tradicional
        Format[θm2, TraditionalForm]
        formato   forma tradicional
        Format[θm3, TraditionalForm]
        formato   forma tradicional
```

Out[•]=

$$\psi \to \tan^{-1}\left(\frac{q_1\, q_3 - q_0\, q_2}{q_0\, q_1 + q_2\, q_3}\right) + \phi_1$$

Out[•]=

$$\theta_1 \to -\tan^{-1}\left(\frac{2\,(q_0\, q_1 - q_2\, q_3)}{\left(2\, q_0^2 + 2\, q_3^2 - 1\right)\cos(\phi_1)}\right)$$

Out[•]=

$$\theta_2 \to -\tan^{-1}\left(\frac{2\,(q_0\, q_1 - q_2\, q_3)}{\left(2\, q_0^2 + 2\, q_3^2 - 1\right)\cos(\phi_2)}\right)$$

Out[•]=

$$\theta_3 \to -\tan^{-1}\left(\frac{2\,(q_0\, q_1 - q_2\, q_3)}{\left(2\, q_0^2 + 2\, q_3^2 - 1\right)\cos(\phi_3)}\right)$$

Cuaterniones de rotación

```
In[•]:= Q1 = ToVector[Simplify[RotationQuaternion[ϕ1, {0, 0, 1}] **
                  simplifica
          RotationQuaternion[-θ1, {1, 0, 0}] ** RotationQuaternion[ψ - ϕ1, {0, 0, 1}]]];
        Q2 = ToVector[Simplify[RotationQuaternion[ϕ2, {0, 0, 1}] **
                  simplifica
          RotationQuaternion[-θ2, {1, 0, 0}] ** RotationQuaternion[-ϕ2, {0, 0, 1}]]];
        Q3 = ToVector[Simplify[RotationQuaternion[ϕ3, {0, 0, 1}] **
                  simplifica
          RotationQuaternion[-θ3, {1, 0, 0}] ** RotationQuaternion[-ϕ3, {0, 0, 1}]]];
```

In[ ]:=
```
ToString[MatrixForm[Q1], TraditionalForm]
ToString[MatrixForm[Q2], TraditionalForm]
ToString[MatrixForm[Q3], TraditionalForm]
```
convierte ··· forma de matriz forma tradicional

Out[ ]=
$$\begin{pmatrix} \cos\left(\frac{\Theta_1}{2}\right)\cos\left(\frac{\psi}{2}\right) \\ \sin\left(\frac{\Theta_1}{2}\right)\left(-\cos\left(\frac{1}{2}(\psi-2\phi_1)\right)\right) \\ \sin\left(\frac{\Theta_1}{2}\right)\sin\left(\frac{1}{2}(\psi-2\phi_1)\right) \\ \cos\left(\frac{\Theta_1}{2}\right)\sin\left(\frac{\psi}{2}\right) \end{pmatrix}$$

Out[ ]=
$$\begin{pmatrix} \cos\left(\frac{\Theta_2}{2}\right) \\ \sin\left(\frac{\Theta_2}{2}\right)(-\cos(\phi_2)) \\ -\sin\left(\frac{\Theta_2}{2}\right)\sin(\phi_2) \\ 0 \end{pmatrix}$$

Out[ ]=
$$\begin{pmatrix} \cos\left(\frac{\Theta_3}{2}\right) \\ \sin\left(\frac{\Theta_3}{2}\right)(-\cos(\phi_3)) \\ -\sin\left(\frac{\Theta_3}{2}\right)\sin(\phi_3) \\ 0 \end{pmatrix}$$

Ángulos de la mano (Cuaterniones de rotación)

In[ ]:=
```
Q = {q0, q1, q2, q3};
```

In[ ]:=
```
ToString[Q, TraditionalForm]
```
convierte a c··· forma tradicional

Out[ ]=
$$\{q_0, q_1, q_2, q_3\}$$

In[ ]:=
```
ψq = Solve[Q1[[4]] / Q1[[1]] == Q[[4]] / Q[[1]], ψ][[1, 1]] /. c1 → 0;
Θq1 = Solve[Simplify[Q1[[2]] / Q1[[1]] == Q[[2]] / Q[[1]]] /. ψq, Θ1][[1, 1]] /. c1 → 0;
Θq2 = Solve[Q2[[2]] / Q2[[1]] == Q[[2]] / Q[[1]], Θ2][[1, 1]] /. c1 → 0;
Θq3 = Solve[Q3[[2]] / Q3[[1]] == Q[[2]] / Q[[1]], Θ3][[1, 1]] /. c1 → 0;
```
resuelve
resue·· simplifica
resuelve
resuelve

In[ ]:=
```
Format[ψq, TraditionalForm]
Format[θq1, TraditionalForm]
Format[θq2, TraditionalForm]
Format[θq3, TraditionalForm]
```

Out[ ]=
$$\psi \to 2 \tan^{-1}\left(\frac{q_3}{q_0}\right)$$

Out[ ]=
$$\theta_1 \to -2 \tan^{-1}\left(\frac{q_1}{q_3 \sin(\phi_1) + q_0 \cos(\phi_1)}\right)$$

Out[ ]=
$$\theta_2 \to -2 \tan^{-1}\left(\frac{q_1}{q_0 \cos(\phi_2)}\right)$$

Out[ ]=
$$\theta_3 \to -2 \tan^{-1}\left(\frac{q_1}{q_0 \cos(\phi_3)}\right)$$

Simulación

In[19]:=
```
Manipulate[
 width = 0.3;
 R1 = EulerMatrix[{ϕ1 °, -θ1 °, (ψ - ϕ1) °}, {3, 1, 3}];
 R2 = EulerMatrix[{ϕ2 °, -θ2 °, -ϕ2 °}, {3, 1, 3}];
 R3 = EulerMatrix[{ϕ3 °, -θ3 °, -ϕ3 °}, {3, 1, 3}];
 ax1 = {0.5, 0, 0}.EulerMatrix[{(ψ - ϕ1) °, 0, 0}, {3, 2, 1}];
 ax2 = {0.5, 0, 0}.EulerMatrix[{-ϕ2 °, 0, 0}, {3, 2, 1}].R1 // Simplify;
 ax3 = {0.5, 0, 0}.EulerMatrix[{-ϕ3 °, 0, 0}, {3, 2, 1}].R2.R1 // Simplify;
 f1 = {0, 4, 0}.R1; f2 = {0, 2.5, 0}.R2.R1 + f1; f3 = {0, 2, 0}.R3.R2.R1 + f2;
 Base = Graphics3D[{Green, Cone[{{0, -1.5, 0}, {0, 0, 0}}, width]}];
 F1 = Graphics3D[{Blue, Tube[{{0, 0, 0}, f1}, width]}];
 F2 = Graphics3D[{Blue, Tube[{f1, f2}, width]}];
 F3 = Graphics3D[{Blue, Tube[{f2, f3}, width]}];
 A1 = Graphics3D[{Red, Cylinder[{ax1, -ax1}, width * 1.7]}];
 A2 = Graphics3D[{Red, Cylinder[{f1 + ax2, f1 - ax2}, width * 1.7]}];
```

```
A3 = Graphics3D[{Red, Cylinder[{f2 + ax3, f2 - ax3}, width * 1.7]}];
     gráfico 3D   rojo  cilindro

Show[Base, F1, F2, F3, A1, A2, A3, PlotRange → All,
muestra                          rango de rep···  todo

  ViewPoint → Vista, ImageSize → {310, 365}, ImageMargins → 5, Boxed → False],
  punto de vista     tamaño de imagen          márgenes de imagen   rodead···  falso

Style["Dedo genérico", Bold, Large],
estilo                 negrita  grande

Delimiter, Style["Ángulo de Abducción y Aducción", Bold, Medium],
delimitador  estilo                                  negrita  tamaño medio

{{ψ, 0, "ψ"}, -27, 15, 1},
Delimiter, Style["Ángulo de Flexión y Extensión", Bold, Medium],
delimitador  estilo                                  negrita  tamaño medio

{{θ1, 0, "θ1"}, 0, 110, 1}, {{θ2, 0, "θ2"}, 0, 135, 1}, {{θ3, 0, "θ3"}, 0, 90, 1},
Delimiter, Style["Ángulo Oblicuo Interfalangico", Bold, Medium],
delimitador  estilo                                  negrita  tamaño medio

{{ϕ1, 14, "ϕ1"}, 0, 14, 1}, {{ϕ2, 7, "ϕ2"}, 0, 7, 1}, {{ϕ3, 7, "ϕ3"}, 0, 7, 1},
Delimiter, Style["Ajustes de control", Bold, Medium],
delimitador  estilo                          negrita  tamaño medio

{Vista, {{-1, -1, 1} → "Perspectiva", Top → "Palmar",
                                        arriba

   Left → "Izquierda", Right → "Derecha"}, ControlType → Setter},
   izquierda           derecha             tipo de control  selector

AppearanceElements → {"HideControlsButton", "ResetButton"},
elementos de apariencia

LabelStyle → Medium,
estilo de etiqueta  tamaño medio

ControlPlacement → Left,
posicionamiento de con···  izquierda

FrameMargins → 0,
márgenes de marco

ContentSize → {320, 375}]
tamaño de contenido
```

Out[19]=

Dedo genérico

Ángulo de Abducción y Aducción

ψ

Ángulo de Flexión y Extensión

θ_1

θ_2

θ_3

Ángulo Oblicuo Interfalangico

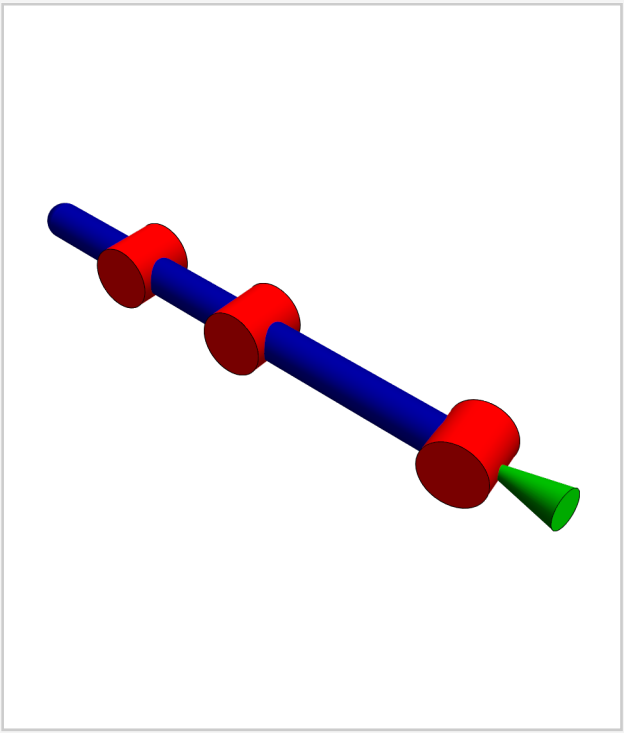
ϕ_1

ϕ_2

ϕ_3

Ajustes de control

Vista | Perspectiva | Palmar | Izquierda | Derec⋱.



Pruebas

In[•]:=
```
angle = 5;
vector = {1, 1, 1};
Qua = ToVector[RotationQuaternion[angle °, vector]] // N;
                                                    valor numérico
qua = {q0 → Qua[[1]], q1 → Qua[[2]], q2 → Qua[[3]], q3 → Qua[[4]]}
Column[
columna
 {m → MatrixForm[RotationMatrix[angle °, vector]] // N, q → MatrixForm[M] /. qua}]
      forma de mat·· matriz de rotación                 valor··· forma de matriz
Column[{q → ψ / ° /. ψq, m → ψ / ° /. ψm} /. qua /. ϕ1 → oblicue °]
columna
Column[{q → θ1 / ° /. θq1, m → θ1 / ° /. θm1} /. qua /. ϕ1 → oblicue °]
columna
```

Out[•]=
$$\{q_0 \to 0.999048,\ q_1 \to 0.0251837,\ q_2 \to 0.0251837,\ q_3 \to 0.0251837\}$$

Out[•]=
$$m \to \begin{pmatrix} 0.997463 & -0.049051 & 0.0515878 \\ 0.0515878 & 0.997463 & -0.049051 \\ -0.049051 & 0.0515878 & 0.997463 \end{pmatrix}$$
$$q \to \begin{pmatrix} 0.997463 & -0.049051 & 0.0515878 \\ 0.0515878 & 0.997463 & -0.049051 \\ -0.049051 & 0.0515878 & 0.997463 \end{pmatrix}$$

Out[•]=
$$q \to 2.88797$$
$$m \to -43.556$$

Out[•]=
$$q \to -2.88797$$
$$m \to -2.81529$$

In[•]:=
```
flexion = 15;
oblicue = 0;
Qua = Q1 /. ψ → 14 ° /. ϕ1 → oblicue ° /. θ1 → flexion ° // N;
                                                    valor numérico
qua = {q0 → Qua[[1]], q1 → Qua[[2]], q2 → Qua[[3]], q3 → Qua[[4]]}
Column[{q → ψ / ° /. ψq, m → ψ / ° /. ψm} /. qua /. ϕ1 → oblicue °]
columna
Column[{q → θ1 / ° /. θq1, m → θ1 / ° /. θm1} /. qua /. ϕ1 → oblicue °]
columna
```

Out[•]=
$$\{q_0 \to 0.984055,\ q_1 \to -0.129553,\ q_2 \to 0.0159071,\ q_3 \to 0.120827\}$$

Out[•]=
$$q \to 14.$$
$$m \to 14.$$

Out[•]=
$$q \to 15.$$
$$m \to 15.$$