UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

IMPLEMENTACIÓN DE RUTINAS DE MOVIMIENTO APLICADOS A ROBOTS MÓVILES MEDIANTE TELEOPERACIÓN BASADA EN RECONOCIMIENTO GESTICULAR

Trabajo de titulación previo a la obtención del

Título de Ingeniero en Electrónica

AUTORES: ALEX GEOVANNY HOLGUÍN NAREA

MAGNO EMILIO ARREAGA VALERIANO

TUTOR: ING. ORLANDO GIOVANNI BARCIA AYALA, MSc

Guayaquil – Ecuador

2025 – 2026

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Alex Geovanny Holguín Narea con documento de identificación N° 0944248574 y Magno Emilio Arreaga Valeriano con documento de identificación N° 0955926688; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Guayaquil, 30 de enero del año 2026.
Atentamente,

Alex Geovanny Holguín Narea
0944248574

Magno Emilio Arreaga Valeriano
0955926688

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Alex Geovanny Holguín Narea con documento de identificación N° 0944248574 y Magno Emilio Arreaga Valeriano con documento de identificación N° 0955926688, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "Implementación de rutinas de movimiento aplicados a robots móviles mediante teleoperación basada en reconocimiento gesticular.", el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 30 de enero del año 2026.

Atentamente,

Alex Geovanny Holguín Narea
0944248574

Magno Emilio Arreaga Valeriano
0955926688

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Orlando Giovanni Barcia Ayala con documento de identificación N° 1309445714, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: Implementación de rutinas de movimiento aplicados a robots móviles mediante teleoperación basada en reconocimiento gesticular., realizado por Alex Geovanny Holguín Narea con documento de identificación N° 0944248574 y Magno Emilio Arreaga Valeriano con documento de identificación N° 0955926688, obteniendo como resultado final el trabajo de titulación bajo la opción de Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 30 de enero del año 2026.

Atentamente,

Ing. Orlando Giovanni Barcia Ayala, MSc
1309445714

DEDICATORIA

Dedico este trabajo, con profundo amor y gratitud, a mis padres, Kleber Holguín y Elizabeth Narea. Gracias por ser mi cimiento y mi horizonte, por brindarme su mano incluso cuando mis sueños parecían inalcanzables y por otorgarme la libertad de intentar lo imposible, todo lo que he logrado es un reflejo de su apoyo incondicional y de su fe inquebrantable en mí.

A los amigos que caminaron a mi lado y a las personas que el destino cruzó en mi camino durante esta etapa; cada palabra de aliento fue el combustible necesario para seguir adelante. De manera especial, rindo tributo a quienes ya no están físicamente, pero cuya confianza en mis capacidades sigue resonando en cada paso que doy.

Mi más sincero agradecimiento a los docentes que marcaron mi formación, especialmente a mi tutor, el Ingeniero Orlando Barcia. Gracias por ser de los primeros en creer en mi potencial desde mi ingreso a la universidad; mantener en alto las expectativas y la confianza que depositaron en mí ha sido, y será siempre, mi mayor compromiso.

Finalmente, me dedico esto a mí mismo. A la versión de mí que entró a las aulas con una idea clara y no la soltó jamás, y a todas las versiones de mi ser que transitaron esta carrera. Honro mi crecimiento, mi perseverancia y la evolución personal que me permite hoy, con orgullo, cerrar este ciclo.

Alex Geovanny Holguín Narea

DEDICATORIA

Dedico este trabajo de titulación a mis padres, por su apoyo incondicional, por el esfuerzo y sacrificio que han realizado a lo largo de mi formación académica y personal, y por inculcarme valores que me han permitido perseverar y no rendirme antes las dificultades.

A mi familia, por estar presente en cada etapa de mi vida, brindándome ánimo, comprensión y motivación para seguir adelante en el cumplimiento de mis metas.

De manera especial a mis abuelos, por su cariño, consejos y apoyo constante, que han sido un pilar fundamental en mi crecimiento personal y académico.

También, dedico a este trabajo a todas las personas que de una u otra forma formaron parte de este proceso, aportando con palabras de aliento y apoyo durante el desarrollo de este proyecto.

Magno Emilio Arreaga Valeriano

AGRADECIMIENTO

Agradezco profundamente a mi madre, por su amor incondicional, por confiar en mí y por brindarme el apoyo necesario para culminar mis estudios universitarios. Gracias a ella, este logro es posible.

A mi familia, por su respaldo constante y por acompañarme en cada etapa de mi formación académica, motivándome a seguir adelante incluso en los momentos más difíciles.

A los docentes de la Carrera de Electrónica y Automatización de la Universidad Politécnica Salesiana, sede de Guayaquil, por los conocimientos impartidos y por contribuir a mi formación profesional a lo largo de la carrera.

Un agradecimiento especial a mi tutor de titulación, por su guía, orientación y apoyo durante el desarrollo de este proyecto, aportando con su experiencia y conocimientos para alcanzar los objetivos planteados.

Le agradezco a mi compañero de tesis Alex Geovanny Holguín Narea, por el trabajo en equipo, el compromiso y el apoyo brindado durante el desarrollo de este trabajo, lo cual permitió culminar satisfactoriamente esta etapa académica.

Magno Emilio Arreaga Valeriano

AGRADECIMIENTO

Agradezco infinitamente a mi padre y a mi madre por serlo todo para mí, por haberme sostenido a pesar de cualquier pronóstico y por darme la mano siempre que decidí perseguir lo difícil, permitiéndome intentar lo imposible y dándome la libertad de fallar hasta lograrlo, porque es solo gracias a ellos que hoy soy todo lo que he podido alcanzar.

A mi amigo y compañero de tesis, Magno Emilio Arreaga Valeriano, por caminar a mi lado desde casi el inicio de esta carrera, compartiendo el peso de este proceso y demostrando que la lealtad es el mejor apoyo para llegar a la meta.

A los amigos que el camino me regaló, a las personas que conocí en cada etapa de este periodo y a quienes ya no están presentes, pero que alguna vez confiaron en mí y me recordaron que era capaz de lograr cualquier cosa que me propusiera, dejando una huella imborrable en mi memoria.

A los ingenieros y docentes que siempre me animaron a no rendirme, especialmente a mi tutor, el Ingeniero Orlando Barcia, por ser de los primeros en creer en aquel estudiante que recién ingresaba a la universidad, y por motivarme a mantener siempre en alto las expectativas y la confianza que cada maestro depositó en mi capacidad.

Por último, agradezco a quien en algún momento se enorgulleció de mí, a quien me hizo esforzarme por ser quien soy y me enseñó a amar mis sueños y aquello a lo que me dedico, a quien me dijo con convicción que podría lograrlo todo y que, al final, todo en esta vida tiene solución

Alex Geovanny Holguín Narea

RESUMEN

AÑO	ALUMNOS	DIRECTOR DEL PROYECTO	TEMA DE TRABAJO DE TITULACIÓN
2026	ALEX GEOVANNY HOLGUÍN NAREA MAGNO EMILIO ARREAGA VALERIANO	ING. ORLANDO GIOVANNI BARCIA AYALA, MSc	IMPLEMENTACIÓN DE RUTINAS DE MOVIMIENTO APLICADOS A ROBOTS MÓVILES MEDIANTE TELEOPERACIÓN BASADA EN RECONOCIMIENTO GESTICULAR.

Este trabajo Titulación tiene como objetivo implementar rutinas de movimiento en un robot móvil mediante la teleoperación basado en reconocimiento gesticula, usando el sensor Leap Motion. La propuesta busca generar una alternativa a los métodos de control tradicionales, los cuales requieren interfaces físicas y habilidades técnicas específicas, limitando su accesibilidad y facilidad de uso.

El proyecto contempla la electrónica de los robots el desarrollo de comunicación entre el sensor y el control. Mediante algoritmos de procesamiento y reconocimiento de gesto, los movimientos de la mano del usuario son interpretados y convertidos en comandos que permiten controlar el desplazamiento del robot en tiempo real. Este enfoque favorece la interacción humano-robot más natural.

La metodología aplicada incluye las etapas de diseño, programación, integración y pruebas del sistema, con el fin de evaluar la capacidad del sensor para capturar gestos de forma confiable, el proyecto contribuye al fortalecimiento del aprendizaje en la robótica, además propones una solución tecnológica para futuras aplicaciones en el ámbito industrial y en sistemas de asistencia para personas con discapacidad motriz.

Palabras claves: Teleoperación, Control gestual, Interacción humano-robot.

ABSTRACT

YEAR	STUDENTS	PRJ. DIRECTOR	SUBJECT
2026	ALEX GEOVANNY HOLGUÍN NAREA MAGNO EMILIO ARREAGA VALERIANO	ING. ORLANDO GIOVANNI BARCIA AYALA, MSc	IMPLEMENTATION OF MOVEMENT ROUTINES APPLIED TO MOBILE ROBOTS THROUGH TELEOPERATING BASED ON GESTURE RECOGNITION.

This project aims to implement movement routines in a mobile robot through teleoperation based on gesture recognition, using the Leap Motion sensor. The proposal seeks to generate an alternative to traditional control methods, which require physical interfaces and specific technical skills, limiting their accessibility and ease of use.

The project covers the robot's electronics and the development of communication between the sensor and the control system. Using gesture processing and recognition algorithms, the user's hand movements are interpreted and converted into commands that allow the robot's movement to be controlled in real time. This approach promotes more natural human-robot interaction.

The methodology applied includes the stages of design, programming, integration, and testing of the system. In order to evaluate the sensor's ability to reliably capture gestures, the project contributes to strengthening learning in robotics and proposes a technological solution for future applications in the industrial field and in assistance systems for people with motor disabilities.

Keywords: Teleoperation, Gestural control, Human-robot interaction.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En la actualidad, la robótica móvil se ha convertido en una herramienta de mucha importancia dentro del ámbito industrial y educativo, debido a la capacidad para ejecutar tareas de desplazamiento y exploración de manera teleoperada. Para un correcto funcionamiento, es necesario optar con métodos de control eficiente, fáciles de usar y precisos, que permitan una adecuada interacción en humano-robot.

El control de los robots se realizaba mediante interfaces físicas tales como los controles remotos, consolas o teclados, estos requerían de alguna destreza específica y pueden limitar su uso en algunos entornos. Aquí es donde entra el control gestual y surge como una alternativa innovadora, ya que permiten operar sistemas robóticos a través de movimientos naturales con la mano, mejorando la experiencia de uso y facilitando la interacción humano-robot. El sensor Leap Motion destaca por su capacidad de captar gestos manuales en tiempo real y un nivel de presión alto.

Este trabajo de Titulación tiene como objetivo principal implementar rutina de movimientos en un robot móvil mediante teleoperación basada en reconocimiento gesticular, utilizando el sensor Leap Motion como control. Con esto se abordan aspectos como el diseño de estructura del robot, el desarrollo de la comunicación entre el sensor y el sistema de control, así como la programación de las rotunas de movimiento guardadas en los gestos capturados.

La finalidad de este proyecto es dar a demostrar la viabilidad del control gestual aplicado a robots móviles y dejar como evidencia las ventajas frente a los métodos tradicionales, fortaleciendo el aprendizaje en la robótica y la automatización, dejando como base para futuras prácticas en el entorno educativo.

2. PROBLEMA

Al inicio, la robótica comenzó como obras de teatro en el año 1921, pasando a obras literarias por Issac Asimoc en 1940, que mezclaban maquinas humanoides o móviles con articulaciones parecidas a las humanas, dando un gran avance al campo de investigación hoy en día. En la actualidad, la robótica se ha orientado hacia el desarrollo de sistemas que sean capaces de realizar de manera autónoma las tareas, reduciendo el uso de operadores y logrando un gran aumento en la eficiencia de hasta un 22% (IFR, 2023), desde las industrias hasta el ámbito educativo. Pero un problema constante se basa en que, durante décadas, el uso de robots se ha limitado casi al 90% a interfaces físicas, tales como teclados y consolas de control, que requieren habilidades y técnicas específicas, donde no todos los operarios pueden acceder con facilidad (WIT, 2023).

En general, al abordar el tema del conocimiento sobre sensores que capturan movimiento, tales como el Leal Motion Controller, se limita en el ámbito académico en comparación con el ámbito social. Este sensor, que da la oportunidad de seguir o rastrear de manera precisa los movimientos de las manos y dedos, ha ido cambiando la forma de interactuar con las máquinas, dando más posibilidades de tener control de robots de manera sencilla mediante gestos (Koenig, Rodrigez, & Secoli, 2020). A pesar de tener un gran potencial, no se han realizado muchas investigaciones o desarrollos tecnológicos orientados a robots móviles, dejando sin aprovechar un recurso que podría utilizarse para cambiar muchas de las formas en las que actualmente se operar diversas máquinas y equipos (Mariuxi, Intriago, & García, 2022).

Tras el gran avance acelerado que tiene la robótica en la industria, y otra parte en el entorno académico como en la materia robótica de la Universidad Politécnica Salesiana continúa siendo limitado y ampara de todo el contenido. En Ecuador (2025) por el ministerio de educación, alrededor de 22 unidades educativas están dentro del programa robótica, representando el 0,14% en todo el país (Vallejo, 2023), por lo que los programas de formación técnica e ingeniería aún no cuentan con contenidos sólidos de la robótica, automatización ni tecnologías asociadas con el entorno de visión artificial, manejo de control avanzado o programación de robots industriales. Esta falta educativa ocurre por la falta de docentes capacitados en tecnologías emergentes. Las instituciones educativas no están

actualizando sus mallas curriculares al ritmo de la automatización, causando una falta de conocimiento entre los estudiantes adquieren y las que demanda el sector productivo. Como consecuencia, las empresas dependen de técnicos o capacitación interna para operar robots, ralentizando el desarrollo en esta especialización (Canio & Di, 2019).

En la actualidad, la industria ha avanzado significativamente en automatización y en la incorporación de robots de diversos procesos productivos. Sin embargo, persisten limitaciones importantes cuando se trata de incluir a profesionales con discapacidades motrices en los puestos técnicos para los que se formaron. Según datos del Instituto Nacional de Estadística y Censos, en Ecuador 7 de cada 100 personas presentan algún tipo de dificultad funcional que afecta sus actividades cotidianas (CONADIS, 2025). A pesar de ello, la participación laboral de personas con discapacidad sigue siendo baja: solo cerca del 30% logra acceder a un empleo formal, y una proporción aún menor se desempeña en áreas técnicas o industriales (AS, 2022). Como resultado, muchos ingenieros con formación en robótica, control o automatización terminan siendo ubicados en cargos administrativos, los cuales, aunque dignos, no representan el ámbito profesional para el que se prepararon. Si bien investigaciones previas han demostrado que dispositivos como el sensor Leap Motion permiten controlar robots industriales de marcas como KUKA, ABB o FESTO, su implementación en entornos reales de producción continúa siendo limitada (Moreira, Neto, & Pires, 2013).

Ante este escenario, surge la necesidad de estudiar e implementar un sistema de control mediante gestos capturados por el sensor Leap Motion, empleándolo inicialmente en un robot móvil como fase experimental en las aulas o laboratorios. Esta aproximación permitiría evaluar su funcionamiento, precisión y posibilidades hacia robots industriales más complejos y hacia personas con discapacidad. Con ello se busca aportar una alternativa tecnológica inclusiva que contribuya a reducir barreras laborales y permita que profesionales con discapacidades motrices desempeñen funciones técnicas dentro del sector industrial sin que su condición física limite su desarrollo profesional (Dr. Bradley & S. Duerstock,, 2020).

2.1.IMPORTANCIA DE LA PROBLEMÁTICA

La robótica móvil cumple un rol cada vez más importante dentro del ámbito industrial, educativo y en investigaciones, ya que permite la elaboración de tareas de desplazamiento, inspecciones y exploraciones en distintos entornos. Sin embargo, una de los principales problemas en la operación de estos sistemas recae en el uso de interfaces de control tradicionales, como consolas físicas o controles remotos, los cuales requieren habilidades técnicas específicas y limitan la accesibilidad para ciertos usuarios, especialmente personas con discapacidad motriz.

El uso de sistemas de control gestual representa una alternativa innovadora que permite una interacción de manera natural e intuitiva entre el humano-robot. El sensor Leap Motion facilita la captura de movimientos de la mano con alta precisión y en tiempo real, dando la facilidad de controlar al robot sin contacto físico y reduciendo la dependencia de dispositivos mecánicos. No obstante, la aplicación de este tipo de tecnología en robots móviles aun es limitada, lo que evidencia la necesidad de desarrollar e investigar nuevas soluciones en este campo.

En el ámbito académico, esta problemática se manifiesta en la escasa incorporación de interfaces gestuales y tecnológicas emergentes en los laboratorios de robótica y automatización. La falta de prácticas orientadas en el control avanzado y la interacción humano-robot limita la formación técnica de los estudiantes y su preparación frente a las demandas actuales del sector industrial.

A partir del análisis realizado, el presente proyecto busca aportar al fortalecimiento del aprendizaje practico mediante la implementación de un control de teleoperación gesticular aplicado a un robot móvil. Esta metodología planteada no solo permite a los estudiantes comprender el funcionamiento de nuevos métodos de control, sino que también fomenta el desarrollo de soluciones tecnológicas inclusivas, promoviendo la accesibilidad y ampliando las posibilidades a aplicación de la robótica en entornos reales.

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Implementar rutinas de movimiento de un robot móvil y de un robot humanoide bípedo mediante el uso del sensor Leap Motion, permitiendo el control de su movimiento a través de gestos manuales en tiempo real.

3.2. OBJETIVOS ESPECÍFICOS

- Diseñar la estructura y configuración electrónica del robot móvil.
- Desarrollar la interfaz de comunicación entre el sensor Leap Motion y el sistema de control de los robots.
- Desarrollar la programación de las rutinas de movimientos del robot según los gestos capturados por el sensor.
- Implementar el control del robot a través de los gestos capturados por el sensor Leap Motion.

4. FUNDAMENTOS TEÓRICOS

4.1. Robótica y robótica móvil

La robótica integra principios de mecánica, electrónica y computación para diseñar máquinas capaces de ejecutar tareas de manera autónoma o semiautónoma. En este campo, los robots móviles representan plataformas capaces de desplazarse en su entorno mediante ruedas u otros mecanismos, guiadas por sistemas de navegación, sensores y algoritmos de control. Estos fundamentos permiten comprender los requisitos para que un robot móvil pueda responder de forma adecuada a señales externas, como las generadas por gestos manuales (AER Automation, 2024).

Figura 1
Robots móviles

Nota: Robots móviles diseñados para moverse de forma autónoma *(AER Automation, 2024)*

4.2. Fundamentos de control gestual en sistemas robóticos

El procesamiento de señales gesticulares constituye una etapa fundamental en los sistemas ya que permite controlar un robot mediante movimientos del usuario. Este procedimiento inicia cuando el sensor registra el gesto, capturando parámetros como aceleración, orientación, rotación o variaciones de luz, dependiendo de la tecnología utilizada. Toda información inicial se recibe como datos sin procesar, los cuales son transformados en señales o impulsos eléctricos que el microcontrolador pueda interpretar para su análisis. Debido a que las mediciones pueden verse afectadas por ruido o movimientos involuntarios, el microcontrolador aplica técnicas de filtrado digital, como filtros de Kalman, promedios móviles u otras técnicas de suavizado, con el fin de obtener una

señal más estable y precisa que permita identificar el gesto de manera confiable (Guachamín & Alejandro, 2018).

Posteriormente, los valores procesados se comparan con patrones gestuales previamente programados dentro del sistema. Este proceso de reconocimiento es crucial ya que permite interpretar acciones específicas del operador, como inclinaciones hacia adelante, giros o elevaciones del brazo, asignándoles funciones concretas del robot. De esta manera, el sistema traduce gestos humanos en ordenes claras que puedes ser interpretadas por la electrónica de control. Una vez que el gesto ha sido identificado, el microcontrolador genera las señales en forma de pulsos PWM, para accionar motores DC, servomotores o motores paso a paso (Guachamín & Alejandro, 2018).

Finalmente, el robot ejecuta la orden asignada, cerrando así el ciclo de interacción. En sistemas más avanzados, se incorpora un mecanismo de retroalimentación mediante encoders u otros sensores de posición que permitan verificar la correcta realización del movimiento solicitado por el usuario. Este ciclo continuo de detección, interpretación y corrección asegura un control gestual eficiente, intuitivo y seguro dentro del entorno robótico, además se obtiene un sistema capaz de responder de manera natural a los movimientos del usuario lo que representa un avance significativo en la interpretación hombre-robot (Guachamín & Alejandro, 2018).

4.3. Interacción Humano-Robot (HRI)

La interacción humano robot estudia los medios por los cuales los usuarios se comunican con sistemas robóticos. La efectividad de esta interacción depende de la naturalidad y simplicidad de la interface, por lo que la robótica moderna busca migrar de controles físicos tradicionales hacia modelos de interacción más intuitivos, como voz, gestos o reconocimiento corporal (Bermúdez, 2022).

Figura 2
Interacción Humano-robot

Nota: Humano-Robot *(AER Automation, 2024)*.

4.4.Control gestual

El control gestual consiste en interpretar movimientos de las manos para activar acciones dentro de un sistema, este señala una forma de interacción ha ganado relevancia por su carácter intuitivo y su aplicación en áreas como realidad aumentada, videojuegos y robótica. Los sistemas de control gestual dependen de algoritmos capaces de identificar patrones de movimiento, posiciones y trayectorias en tiempo real (Pastor Bernal, 2023).

Figura 3
Lector del sensor Leap Motion

Nota: Lector de gestos (Cortes Rogriguez, Sandoval, & Trujillo Pascual, 2025)

4.5. Sensores de captura de movimiento

Los sensores de captura de movimiento, como cámaras infrarrojas, dispositivos inerciales o sensores ópticos, permiten digitalizar movimientos del usuario. Estos dispositivos han sido ampliamente utilizados en interfaces naturales debido a su capacidad para interpretar gestos sin contacto físico. En el contexto de la robótica, permiten que un robot responda directamente a señales humanas (Universidad de arte Madri, 2023).

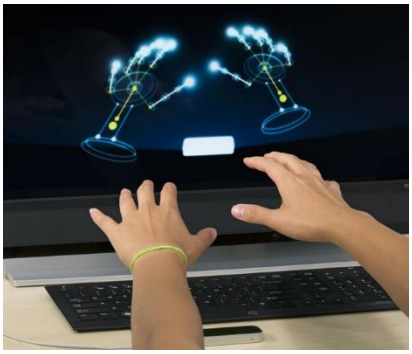
Figura 4
Muestreo del sensor Leap Motion

Nota: Sensor leap motion (TME, 2021)

4.6. Sensor Leap Motion interacción natural

El Leap Motion es un sensor óptico diseñado para rastrear manos y dedos mediante cámaras infrarrojas. Su precisión y bajo costo lo han convertido en un dispositivo popular para proyectos de interacción natural. Weichert et al. (2013) demostraron que su margen de error es lo suficientemente bajo como para utilizarlo en tareas de control fino. Tölgyessy et al. (2020) destacan su desempeño en aplicaciones educativas y de teleoperación (Shukla, 2020).

Figura 5
Vista de la interfaz del sensor Leap Motion



Nota: Muestreo del sensor Leap Motion (Parreño Alvarez, 2021)

4.7. Accesibilidad tecnológica

La accesibilidad tecnológica busca garantizar que todas las personas, independientemente de sus limitaciones físicas, puedan utilizar herramientas digitales. La ONU (2016) señala que las tecnologías deben eliminar barreras de acceso, permitiendo que las personas con discapacidad motriz interactúen con sistemas informáticos y robóticos mediante interfaces adaptadas. En este sentido, los controles gestuales representan una alternativa inclusiva, pues no requieren fuerza física ni manipulación compleja (Friendly Captcha, 2023).

4.8. Interfaz gestual para la manipulación de sistemas robóticos

El artículo indica el desarrollo del interfaz gestual electrónica diseñada para el control remoto de un manipulador robótico de 3 grados de libertad. Su elaboración ofrecer, otra forma accesible de manipular y programar robots sin tener conocimiento avanzado en programación, ayudando tanto a estudiantes como a operadores industriales (Rodríguez Baeza, Vergara Betanc, & Osorio Verde, 2022).

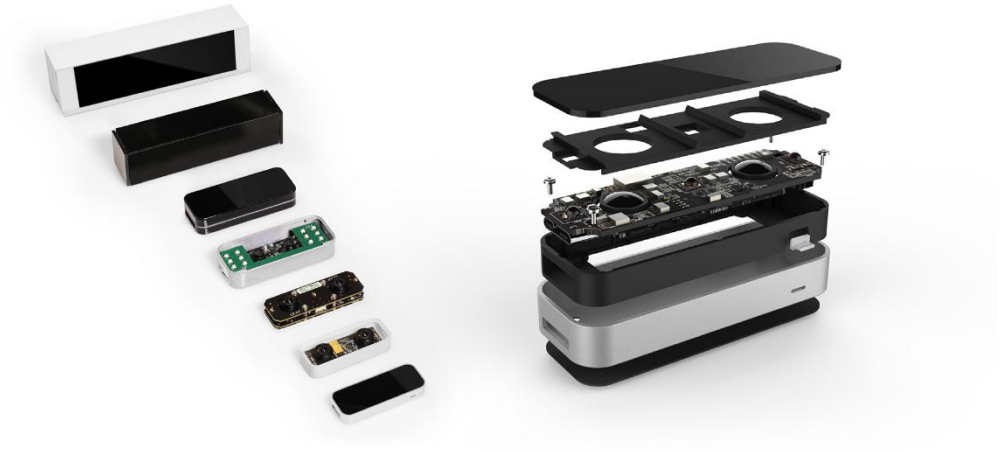
Los resultados mostraron en tiempo real los movimientos de la mano, aunque aún presenta defectos leves por el ruido de los sensores, comunicación bluetooth y movimientos

forzados en servomotores. Aun así, se logró demostrar que es posible manipular un robot de forma intuitiva y sin codificación compleja y extensa, dando paso a nuevas técnicas de programación gesticulares aplicables en educación, investigación e industria. (Rodríguez Baeza, Vergara Betanc, & Osorio Verde, 2022).

4.9. Sensor Leap Motion

El sensor Leap Motion Controller o LMC es un dispositivo óptico especializado en la captura de movimiento de manos y dedos en tres dimensiones sin necesidad de contacto físico. Este fue desarrollado para ofrecer una interfaz natural de interacción humano-computadora mediante gestos, lo que permite controlar sistemas digitales o robots sin manos tradicionales. El sensor utiliza luz infrarroja y cámaras para detectar la posición espacial de los dedos en tiempo real, lo que lo convierte en una herramienta ideal para aplicaciones de reconocimiento gestual y teleoperación (Guna, Jakus, & Sodnik, MDPI, 2021).

Figura 6
Sensor Leap Motion



Nota: El sensor Leap Motion y sus características (KYLEDESING, 2025).

4.9.1. Componentes y estructura del señor Leap Motion

Su estructura es compuesta por tres emisores infrarrojos de dos cámaras infrarrojas que capturan información visual en un volumen sobre el dispositivo. Sus emisores proyectan luz infrarroja y las cámaras registran los reflejos de los objetos en este caso las manos y dedos, permitiendo reconstruir posibilidad la detención de múltiples puntos de la mano simultáneamente en los tres ejes espaciales (X, Y y Z) (Bachmann, Weichert, & Rinkernauer, 2015).

4.9.2. Especificaciones del sensor Leap Motion

Las especificaciones del Leap Motion las cuales influyen de manera directa en su rendimiento son:

- La frecuencia de fotogramas o los frame rate, que son hasta 120 fps, los cuales favorecen el seguimiento de gestos suaves.
- Su campo de visión que tiene aproximadamente 150° * 120°, lo que permite un área considerable para captar gestos manuales sin mover el sensor.
- El rango de detección del sensor puede captar posiciones de manos hasta los 60 a 80 cm por encima del mismo.
- La resolución espacial tal indica su data, puede alcanzar seguimientos submilimétricos, con rendimiento de precisión en detección de posiciones estáticas menores a 0.5 mm.

Estas las especificaciones la cual permiten un control gestual con alta fidelidad para aplicaciones de interacción y robótica (LEAP MOTION ULTRALEAP, 2021).

4.9.3. Velocidad y rendimiento del sensor Leap Motion

El rendimiento del Leap Motion, este puede variar según la configuración del sistema y el entorno. Aunque puede alcanzar hasta 120 fps, investigaciones han observado que en uso practico y del software que procesa la información. Esta alta tasa de captura permite que los gestos se detecten con una mínima latencia, lo que es

fundamental para aplicaciones de control en tiempo real, como en robots móviles o interfaces interactivas (Bird, 2022).

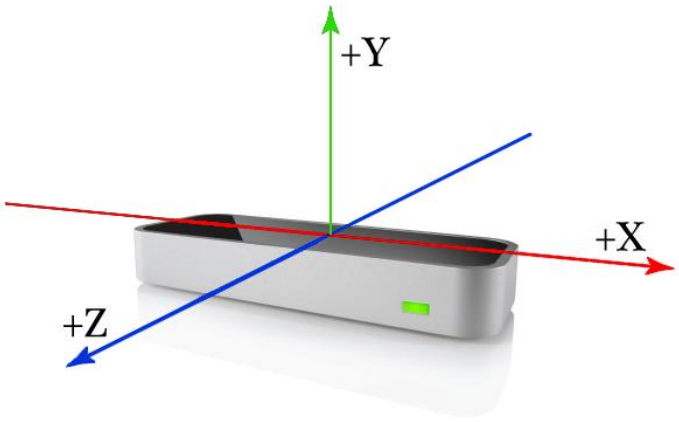
4.9.4. Precisión y confiabilidad del sensor Leap Motion

En los trabajos comparativos con sistemas referentes de alta precisión, el sensor ha mostrado resultados confiables en escenarios estáticos y dinámicos. Sin embargo, la precisión disminuye cuando las manos se alejan del volumen optimo. Este comportamiento sugiere que el rango operativo recomendado para aplicaciones que demandan alta exactitud se encuentra cerca del sensor y dentro su campo visual definido. (RobotShop, 2024)

4.9.5. Limitaciones y campo de interacción del sensor Leap Motion

Este se define como un volumen de interacción ubicado por encima del dispositivo, donde se detectan los movimientos de las manos. Este volumen tiene forma de cono invertido, con una mayor precisión cerca de la superficie del sensor. Algunas limitaciones reportadas incluyen sensibilidad reducida para distinguir gestos cuando las manos se superponen i se encuentran parcialmente fuera del volumen de detención (López, 2021).

Figura 7
Dimensión del sensor Leap Motion



Nota: Ejes del Leap Motion para la detección (ULTRALEAP, 2026).

4.9.6. Software y desarrollo del sensor Leap Motion

Este viene acompañado de un SDK y APIs que permiten programar la lectura de datos de posición, velocidad y gestos en tiempo real. Estas herramientas de desarrollo son compatibles con múltiples lenguajes de programación y plataformas, lo que facilita su integración en proyectos de automatización, robótica o realidad virtual y aumentada (ULTRALEAP, 2025).

4.9.7. Aplicaciones y estudios del sensor Leap Motion

El sensor Leap Motion ha sido utilizado para el reconocimiento de gestos en sistemas de lenguaje de señas, clasificación gestual, entrenamiento con modelos de aprendizaje automático y análisis de movimientos humanos en aplicaciones biomédicas, Su uso va desde interfaces de interacción Humano-Robot o Humano-Computadora hasta control de dispositivos sin contacto físico, haciendo posibles la exploración de nuevas formas de teleoperación robótica y accesibilidad en sistemas automatizados (Kajan, y otros, 2024).

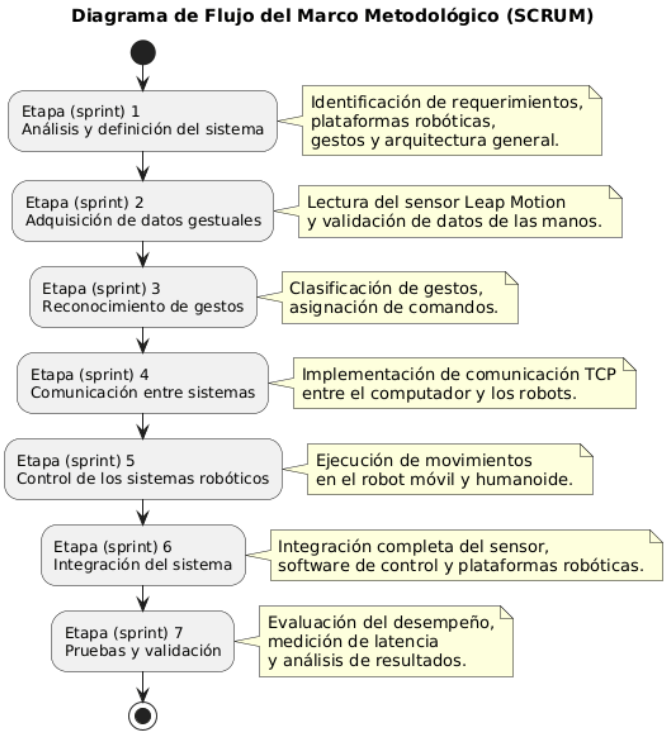
5. MARCO METODOLÓGICO

5.1. Descripción del proyecto

El presente proyecto se orienta a la implementación de rutinas de movimientos en un robot móvil mediante un sistema de teleoperación basado en reconocimiento gesticular, utilizado el sensor Leap Motion como dispositivo principal de captura de movimientos manuales. El objetivo es lograr que los desplazamientos básicos de los robots, tales como avance, retroceso, giros laterales y detención, sean ejecutados de forma precisa a partir de gestos manuales interpretados en tiempo real.

La figura 8 muestra el diagrama de flujo del marco metodológico basado en SCRUM, el cual describe las etapas desarrolladas para la implementación del sistema de teleoperación gesticular.

Figura 8
Diagrama de flujo del marco metodológico basado en SCRUM para el desarrollo del sistema de teleoperación gesticular.



El sistema propuesto está conformado por tres componentes principales: el sensor Leap Motion encargado de la adquisición de gestos, un computador que realiza el

procesamiento, clasificación y generación de comandos, un robot bípedo y un robot móvil que ejecuta las rutinas de movimiento a partir de las órdenes recibidas. La comunicación entre el sistema de reconocimiento gesticular y el robot se establece mediante un canal de transmisión inalámbrico vía WIFI, permitiendo él envió de instrucciones en tiempo real.

Las rutinas de movimiento implementadas permiten controlar el desplazamiento del robot en función de los gestos realizados por el usuario, considerando aspectos como la velocidad de desplazamiento, la dirección, el tiempo de respuesta y la estabilidad dinámica del sistema. Asimismo, se incorporan mecanismos de seguridad y estados de parada que evitan movimientos no deseados y garantizan la integridad del sistema durante la operación.

Finalmente, el proyecto contempla una fase de validación experimental en un entorno de laboratorio, en la cual se evalúa el desempeño del sistema de teleoperación gesticular mediante pruebas funcionales, mediciones de tiempo de respuestas y análisis de precisión en la interpretación de gestos. Esta etapa permite comprobar la viabilidad técnica de las propuestas y su potencial aplicación en entornos educativos, industriales y sistemas de asistencia orientadas a la interacción humano-robot.

5.2. Tipo de metodología

Para el desarrollo del presente proyecto se aplicaron tres enfoques metodológicos complementarios: metodología experimenta, metodología sistemática y metodología ágil Scrum, los cuales permitieron estructurar, ejecutar y validar de manera ordenada el proceso de implementación del sistema de teleoperación gesticular aplicado a robots móviles.

5.2.1. Metodología Experimental

La metodología experimental con el fin de evaluar el comportamiento del prototipo en condiciones reales de operación. Este enfoque permitió realizar pruebas controladas sobre cada uno de los componentes del sistema, tales como la captura de gestos mediante el sensor Leap Motion, la clasificación de los movimientos, la transmisión de comandos y la respuesta dinámica del robot móvil.

Mediante la implementación de múltiples iteraciones se ajustaron parámetros relacionados con la velocidad, dirección, estabilidad y tiempo de respuestas del sistema, permitiendo verificar el correcto funcionamiento de las rutinas de movimiento y garantizar un control seguro y confiable. Este método facilito la validación progresiva del sistema y la corrección de errores detectados durante la fase de pruebas.

5.2.2. Metodología Sistemática

La metodología sistemática permitió organizar el desarrollo del proyecto de forma estructurada y secuencial, estableciendo etapas claramente definidas que abarcaron el análisis, diseño, programación, integración y validación del prototipo.

Esta metodología garantizo una adecuada planificación de las actividades, facilito la identificación temprana de fallas y aseguro la coherencia entre los fundamentos teóricos y la implementación práctica. Además, permito mantener un control ordenado del avance del proyecto y el cumplimiento de los objetivos planteados.

5.2.3. Metodología SCRUM

Como apoyo para el desarrollo del prototipo se empleó la metodología ágil Scrum, con el objetivo de organizar el trabajo técnico de forma iterativa e incremental,

Esta metodología permitió dividir el proyecto en etapas cortas denominadas sprints, en la cual se integró y desarrolló de manera progresiva las distintas funcionalidades del sistema.

Para el ámbito de la electrónica y automatización, Scrum da la facilidad de gestión del desarrollo de software, la integración de hardware y la implementación de algoritmos de control, permitiendo realizar ajustes continuos frente a cambios en los requerimientos técnicos. Asimismo, se fomentó el trabajo colaborativo como integrantes del proyecto, la detención temprana de errores y la mejora continua del sistema, contribuyendo a una implementación eficiente sin comprometer la calidad ni la confiabilidad del prototipo.

5.3.Etapas del Prototipo

5.3.1. Etapa (sprint) 1: Análisis de requisitos

En esta primera etapa se realiza un análisis detallado de los requerimientos técnicos del sistema de teleoperación gesticular, considerando tanto el funcionamiento del sensor Leap Motion como las características mecánicas y de control de las plataformas robóticas a emplear.

Inicialmente, se efectuó un estudio del sensor Leap Motion, analizando sus principios de funcionamientos, campo de visión, rango de detención, frecuencia de muestreo y las condiciones necesarias para obtener datos confiables. Asimismo, se investigó el uso del kit de desarrollo de software (SDK) oficial y su integración con el lenguaje de programación Python, verificando la correcta detención de las manos, la identificación de la mano izquierda y derecha de parámetros relevantes como posición, orientación y nivel de apertura de la mano.

De manera paralela, se analizó la arquitectura del robot móvil basado en un microcontrolador ESP32, considerando su sistema de control, los módulos de

comunicación, las rutinas de movimiento predefinidas y las limitaciones mecánicas del sistema. En esta fase se definieron los gestos principales que serían utilizados como interfaz de control, estableciendo la correspondencia entre cada gesto y las acciones específicas del robot, tales como avanzar, retroceder, girar a la izquierda, girar a la derecha y detenerse. Para esta definición se consideraron criterios de naturalidad, facilidad de ejecución y baja probabilidad de activación involuntaria.

Adicionalmente, se estableció los requerimientos iniciales para la integración del robot humanoide bípedo, considerando su arquitectura de control basada en el sistema operativo robótico ROS (Robot Operating System). En esta etapa se define los tipos de movimientos básicos a implementar, así como las condiciones necesarias para su control gestual en una etapa posterior del proyecto.

Como resultado de esta etapa se obtuvo la especificación funcional del sistema de control gestual, así como la definición de los gestos, comandos y parámetros iniciales que guiaron el desarrollo de las fases posteriores.

5.3.2. Etapa (sprint) 2: Instalación de la herramienta Leap Motion y pruebas iniciales

En esta etapa se realizó la instalación y configuración del sensor Leap Motion en el computador, así como la preparación del entorno de desarrollo necesario para la adquisición y transmisión de datos gestuales. Para este propósito se instaló el kit de desarrollo de software (SDK) oficial del sensor y se configuraron las librerías correspondientes en el lenguaje de programación Python.

Posteriormente, se desarrollaron scripts de prueba en Python que permitieron establecer la comunicación directa entre el sensor Leap Motion y el computador, verificando la correcta recepción de datos asociados a la posición, orientación, número de dedos extendidos y estado de apertura de la mano en tiempo real. Estas pruebas permitieron confirmar la estabilidad de la conexión, la frecuencia de actualización de los datos y la correcta interpretación de los parámetros proporcionados por el sensor. Para validar inicialmente su funcionamiento del sensor Leap Motion se implementó el código fuente en los anexos figura 35.

Adicionalmente, se implementa rutinas para el procesamiento preliminar y el envío de los datos gestuales hacia los módulos de control mediante comunicación por red, con el objetivo de validar la transmisión de información y preparar la integración posterior con los sistemas robóticos. Durante esta fase se ajustaron umbrales y condiciones de lectura con el fin de minimizar el ruido generado por movimientos involuntarios y garantizar una señal confiable.

Como resultado de esta etapa se obtuvo un sistema funcional de adquisición y transmisión de datos gestuales mediante Python, que permitió validar el correcto funcionamiento del sensor Leap Motion y establecer la base para el desarrollo de las etapas posteriores de reconocimiento de gestos e integración con los robots.

Figura 9
Adquisición de datos gestuales mediante el sensor Leap Motion en Python.

```
35
36      class LeapListener(leap.Listener):  1 usage
37          def on_tracking_event(self, event):
38              hands = event.hands
39
```

5.3.3. Etapa (sprint) 3: Programación del reconocimiento de gestos

En esta etapa se desarrolló la lógica de reconocimiento y clasificación de gestos utilizando el lenguaje de programación Python y el kit de desarrollo de software del sensor Leap Motion. A partir de los datos adquiridos en la etapa anterior, se implementaron algoritmos para la lectura y procesamiento de parámetros relevantes tales como la posición de la mano, el número de dedos extendidos, la identificación de la mano izquierda y derecha, y el nivel de cierre de la mano (grab strength).

Figura 10
Clasificación de estados de la mano para el reconocimiento de gestos robot móvil.

```
83          lg, rg = left.grab_strength, right.grab_strength
84
85          l_open, r_open = lg < 0.4, rg < 0.4
86          l_closed, r_closed = lg > 0.8, rg > 0.8
```

Figura 11
Clasificación de estados de la mano para el reconocimiento de gestos robot humanoide.

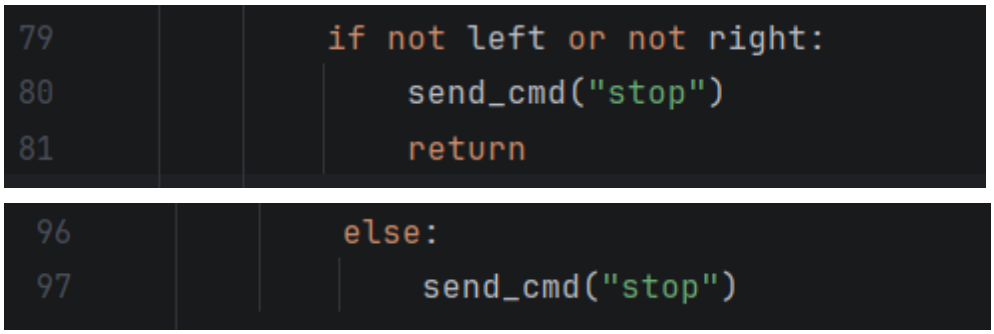
```
57          lg = left.grab_strength
58          rg = right.grab_strength
59
60          print(f"L:{lg:.2f}  R:{rg:.2f}")
61
62          left_open = lg < OPEN_TH
63          right_open = rg < OPEN_TH
64          left_closed = lg > CLOSED_TH
65          right_closed = rg > CLOSED_TH
66
67          if left_open and right_closed:
68              send_cmd("forward")
69          elif left_closed and right_open:
70              send_cmd("back")
71          elif left_closed and right_closed:
72              send_cmd("right")
73          elif left_open and right_open:
74              send_cmd("left")
75          else:
76              send_cmd("stop")
77
```

Se establecieron umbrales de activación para diferenciar entre estados de mano abierta y cerrada, así como condiciones lógicas para la identificación de gestos

específicos mediante el uso de estructuras condicionales. De esta manera, se definieron gestos básicos asociados a comandos de control, tales como avance, retroceso, giro a la izquierda, giro a la derecha y detención.

Adicionalmente, se incorporó un mecanismo de temporización (cooldown) con el fin de evitar la generación repetitiva de comandos ante movimientos involuntarios o fluctuaciones en la señal proporcionada por el sensor. Este mecanismo permitió mejorar la estabilidad del sistema y garantizar una interpretación más confiable de los gestos realizados por el usuario.

Figura 12
Generación automática del comando de detención ante estados gestuales no validos



Durante esta etapa se realizaron múltiples pruebas funcionales para verificar la correcta detención de los gestos, la estabilidad del reconocimiento y la coherencia entre los movimientos realizados por el usuario y los comandos generados por el sistema.

Como resultados de esta etapa se obtuvo un módulo funcional de reconocimientos gestual capaz de transformar los movimientos de las manos en instrucciones digitales estables y confiables, que sirvió como base para el desarrollo de la comunicación con los sistemas robóticos en las etapas posteriores.

Con el fin de establecer una correspondencia clara entre los movimientos manuales del usuario y las acciones de los sistemas robóticos, se definió un conjunto de gestos básicos utilizando ambas manos como entrada del sistema. El reconocimiento de gestos se realizó a partir del parámetro grab strength

proporcionado por el sensor Leap Motion, el cual permite identificar el estado de apertura o cierre de cada mano.

El sistema fue diseñado bajo un esquema de seguridad por defecto, en el cual el comando de detención se mantiene activo cuando no se detectan dos manos simultáneamente, cuando la combinación de gestos no es válida o cuando se pierde el seguimiento del sensor, evitando desplazamiento no controlados del robot.

Debido a las diferencias cinemáticas entre el robot móvil y el robot humanoide bípedo, se definieron dos esquemas de mapeo de gestos, uno para cada plataforma, permitiendo una adaptación adecuada del sistema de control gestual a las características de cada robot.

Tabla 1
Definición de gestos para el control del robot móvil.

Mano izquierda	**Mano derecha**	**Comando generado**	**Acción del robot**
Abierta	Abierta	Forward	Avance
Cerrada	Cerrada	Back	Retroceso
Abierta	Cerrada	Left	Giro a la izquierda
Cerrada	Abierta	Right	Giro a la derecha
Combinación no valida / ausencia de manos	N/D	stop	Detención

Figura 13
Comando capturado para movimiento hacia adelante para el robot móvil.

Tabla 2
Definición de gestos para el control de robot humanoide bípedo.

Mano izquierda	Mano derecha	Comando generado	Acción del robot
Abierta	Abierta	Left	Giro a la izquierda
Cerrada	Cerrada	Right	Giro a la derecha
Abierta	Cerrada	Forward	Avance
Cerrada	Abierta	Back	Retroceso
Combinación no valida / ausencia de manos	N/D	stop	Detención

Figura 14
Comando capturado para movimiento hacia la derecha para robot humanoide.

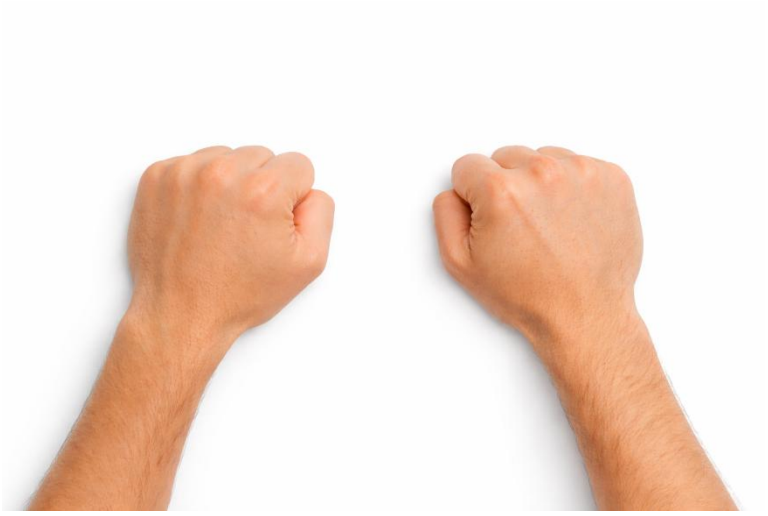


Figura 15
Detención de dirección de movimiento a partir de gestos manuales.

```
48      # ---------- DIRECCIÓN ----------
49      def detect_direction(self, left_state, right_state):  1 usage
50          if left_state == "OPEN" and right_state == "CLOSED":
51              return "ADELANTE"
52          if left_state == "CLOSED" and right_state == "OPEN":
53              return "ATRÁS"
54          if left_state == "CLOSED" and right_state == "CLOSED":
55              return "DERECHA"
56          if left_state == "OPEN" and right_state == "OPEN":
57              return "IZQUIERDA"
58          return None
```

5.3.4. Etapa (sprint) 4: Desarrollo de la comunicación entre el ordenador y el robot

En esta etapa se implementó el sistema de comunicación entre el computador y el robot móvil mediante el uso de sockets TCP, permitiendo la transmisión en tiempo real de los comandos generados por el módulo de reconocimiento gestual. Para este propósito se desarrolló un cliente de comunicación en Python que establece una conexión directa con el servidor implementando en el microcontrolador ESP32, utilizando una dirección IP y un puerto de comunicación previamente configurados.

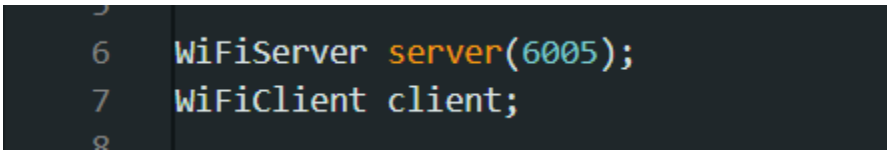
Figura 16
Elaboración del proyecto para la comunicación.

El módulo de comunicación fue diseñado para enviar de manera continua los comandos de movimientos generados a partir de los gestos reconocidos, tales como avance, retroceso, giros laterales y detención. La transmisión de datos se realiza utilizando el protocolo TCP, garantizando una comunicación confiable y orientación a conexión entre el computador y el sistema de control del robot.

Figura 17
Implementación del cliente TCP para la comunicación con el robot humanoide.

```
22    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
23    sock.settimeout(0.1)
24    sock.connect((ROBOT_IP, PORT))
```

Figura 18
Implementación del servidor TCP en el microcontrolador ESP32 para la comunicación con el sistema de control robot móvil.



Durante esta etapa se implementaron mecanismo para la inicialización de la conexión, él envió de cadenas de texto correspondientes a los comandos de control y la recepción de mensajes de confirmación por parte del robot, permitiendo verificar la correcta entrega de cada instrucción. Asimismo, se realizaron pruebas funcionales para evaluar la estabilidad de enlace de comunicación y el tiempo de respuesta del sistema ante cambios en los gestos realizados por el usuario.

Como resultado de esta etapa se obtuvo un canal de comunicación funcional entre el computador y el robot móvil, capaz de transmitir de forma confiable los comandos generados por el sistema de reconocimiento gestual y servir como base para la integración completa del sistema de teleoperación.

5.3.5. Etapa (sprint) 5: Configuración del robot móvil y humanoide y desarrollo de rutinas de movimiento

En esta etapa se realizó la configuración del sistema de control del robot móvil basado en el microcontrolador ESP32, así como el desarrollo de las rutinas de movimiento necesarias para la ejecución de los comandos generados por el sistema de teleoperación gesticular. Para este propósito se implementó un servidor de comunicación inalámbrica utilizando el módulo WIFI integrado del ESP32, permitiendo la recepción de comandos enviados desde el computador mediante el protocolo TCP.

Inicialmente, se configuraron los pines digitales y de modulación por ancho de pulso (PWM) correspondientes al sistema de accionamiento de los motores, definiendo las señales de habilitación y dirección para cada uno de los actuadores.

Posteriormente, el microcontrolador fue conectado a una red inalámbrica local y se habilito un servidor TCP en un puerto específico para la recepción de instrucciones provenientes del módulo de comunicación desarrollado en Python.

Una vez establecida la comunicación, se implementó un módulo de procesamiento de comandos encargados de interpretar las cadenas de texto recibidas y asociadas a rutinas específicas de movimientos, tales como avance, retroceso, giro a la izquierda, giro a la derecha y detención. Cada comando recibido activa una función correspondiente que controla el sentido de giro y la velocidad de los motores mediante señales digitales y modulación PWM.

Figura 19
Procesamiento de comandos recibidos y activación de rutinas de movimiento.

```
88    if (cmd == "forward") moveForward();
89    else if (cmd == "back") moveBackward();
90    else if (cmd == "left") turnLeft();
91    else if (cmd == "right") turnRight();
92    else if (cmd == "stop") stopMotors();
```

Adicionalmente, se incorporó un sistema de seguridad basado en temporización que permite detener automáticamente el robot en caso de no recibir nuevos comandos dentro de un intervalo de tiempo definido, evitando desplazamiento no deseados ante pérdidas de comunicación. Asimismo, se implementó un mecanismo de confirmación (acknowledgment) para notificar al computador la correcta recesión de cada comando enviado.

Durante esta etapa se realizaron pruebas funcionales de cada rutina de movimiento, verificando la correcta ejecución de los desplazamientos y la respuesta del robot móvil ante los comandos transmitidos por el sistema de control gestual.

Como resultado de esta obtuvo un sistema funcional de control del robot móvil capaz de ejecutar de manera confiable las rutinas de movimientos a partir de los comandos recibidos por la comunicación inalámbrica, estableciendo la base para la integración final del sistema de teleoperaciones.

5.3.6. Etapa (sprint) 6: Implementación del sistema de gestos y el robot móvil e humanoide.

En esta etapa se realizó la integración completa entre e sistema de reconocimiento gestual basado en el sensor Leap Motion, el módulo de procesamiento desarrollado en Python y los sistemas robóticos correspondientes al robot móvil y al robot humanoide bípedo. Esta integración permitió implementar un sistema de teleoperaciones gesticular capaz de generar comandos de control a partir de los movimientos manuales del usuario y transmitirlos en tiempo real hacia las plataformas robóticas mediante comunicación inalámbrica.

El módulo de control en Python fue diseñado para recibir los eventos de seguimiento proporcionados por el sensor Leap Motion, proceder los parámetros gestuales de ambas manos y generar los comandos de movimiento de acuerdo con los esquemas de mapeo definidos para cada robot. El reconocimiento de gestos se realizó utilizando el parámetro grab strength, permitiendo identificar los estados de apertura y cierre de cada mano y establecer las combinaciones necesarias para la generación de instrucciones de control.

Con el fin de garantizar un funcionamiento seguro del sistema, se implementó un estado de detención por defecto, en el cual el comando de parada se mantiene activo cuando no se detectan las manos simultáneamente o individual, cuando la combinación de gestos no es válida o cuando se pierde el seguimiento del sensor. Este mecanismo permitió evitar desplazamiento no controlados ante errores de reconocimiento o interrupciones en la señal.

Una vez generado cada comando, este fue transmitido desde el computador hacia el sistema robótico correspondiente mediante comunicación inalámbrica utilizando sockets TCP. Para el robot móvil, el módulo de control establecido una conexión directa con el servidor implementado en ele microcontrolador ESP32, permitiendo el envío de instrucciones de movimientos y la recepción de mensaje de confirmación. En el caso del robot humanoide bípedo, el sistema fue integrado mediante la plataforma ROS, la cual fue utilizada como middleware de comunicación

y gestión de nodos para la recepción de comandos generados por el módulo de control en Python. Cada instrucción recibida activa un nodo encargado de ejecutar acciones de movimiento previamente programadas en el sistema del robot, mientras que el control de bajo nivel de la locomoción es gestionado internamente por el propio robot humanoide.

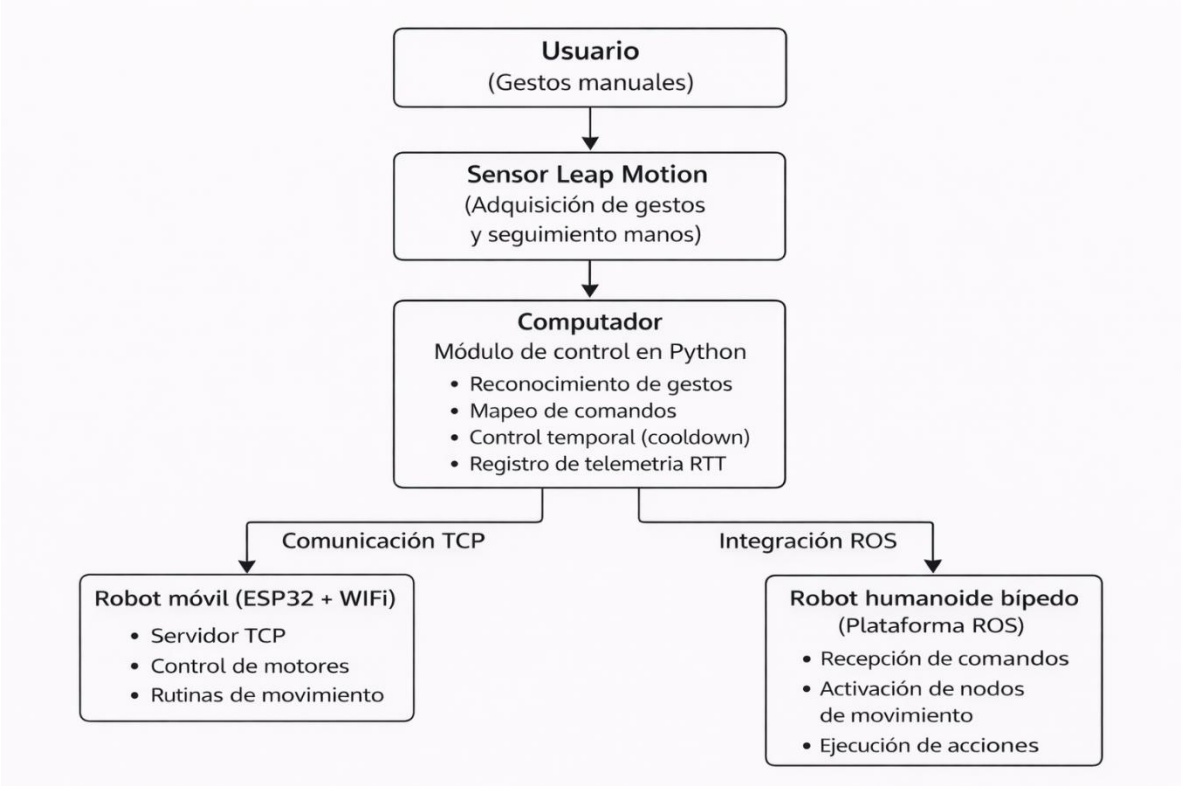
Adicionalmente, se incorporó un sistema de control temporal (cooldown) que limita la frecuencia de envío de comandos, evitando la generación repetitiva de instrucciones ante fluctuaciones rápidas en la señal gestual. Asimismo, se implementó un sistema de telemetría que permite calcular el tiempo de ida y vuelta (round-trip time, RTT) de cada comando enviado, registrando automáticamente los datos de tiempo, comando y respuesta del robot en un archivo en formato CSV para su posterior análisis.

Durante esta etapa se realizaron pruebas integrales del sistema completo, verificando la correcta generación de comandos a partir de los gestos manuales, la transmisión confiable de los datos mediante comunicación TCP y la respuesta adecuada de los sistemas robóticos ante las instrucciones recibidas.

Como resultado de esta etapa se obtuvo un sistema integrado de teleoperación gesticular capaz de controlar en tiempo real el desplazamiento del robot móvil y establecer la base para la integración con el robot humanoide bípedo, preparando el prototipo para la etapa final de pruebas y validación.

Figura 20

Arquitectura general del sistema de teleoperación gesticular



5.3.7. Etapa (sprint) 7: Pruebas finales y validación del prototipo

En esta etapa se realizaron las pruebas finales del sistema de teleoperación gesticular con el fin de validar su correcto funcionamiento, evaluar su desempeño temporal y verificar la confiabilidad del reconocimiento de gestos y de la comunicación inalámbrica con los sistemas robóticos.

Las pruebas experimentales se llevaron a cabo en un entorno controlado de laboratorio, utilizando el sensor Leap Motion para la adquisición de gestos manuales, el módulo de control desarrollado en Python para el procesamiento de señales y generación de comandos, y los sistemas robóticos correspondientes al robot móvil y al robot humanoide bípedo como plataforma de prueba.

Inicialmente, se evaluó la correcta ejecución de los gestos definidos en la etapa de reconocimiento, verificando que cada combinación manual generara el comando

correspondiente y que el robot ejecutara el movimiento separado. Se realizo múltiples repeticiones de cada gesto con el fin de analizar la estabilidad del reconocimiento y la coherencia entre los movimientos realizados por el usuario y las acciones ejecutadas por los robots.

Posteriormente, se evaluó el desempeño temporal del sistema de comunicación mediante el análisis del tiempo de ida y vuelta (round-trip time, RTT) de los comandos transmitidos desde el computador hacia el robot móvil. Para este propósito, se utilizó el sistema de telemetría implementado en el módulo de control de Python, el cual registro automáticamente en un archivo de formato CSV la marca temporal, el comando enviado, la respuesta recibida y el tiempo total de comunicación correspondiente a cada instrucción.

Adicionalmente, se verifico el correcto funcionamiento de los mecanismos de seguridad implementados en el sistema, particularmente el estado de detención por defecto y el sistema de parada automática ante la ausencia de comandos dentro de un intervalo de tiempo determinado, Estas pruebas permitieron garantizar que el sistema evitara desplazamientos no controlados ante perdidas de comunicación o errores de reconocimiento.

Finalmente, se realizaron pruebas integrales de operación continua, evaluando la estabilidad general del sistema, la capacidad de respuestas ante cambios rápidos de gestos y la confiabilidad de la comunicación inalámbrica durante periodos prolongados de funcionamiento.

Como resultado de esta etapa se validó el correcto funcionamiento del sistema de teleoperaciones gesticular, demostrando su capacidad para controlar de manera confiable los desplazamientos del robot móvil y robot humanoide bípedo.

6. RESULTADOS

Los resultados experimentales obtenidos durante la validación del sistema de teleoperación gesticular desarrollado. Las pruebas realizadas tuvieron como objetivo evaluar el desempeño del reconocimiento de gesto, la confiabilidad de la comunicación inalámbrica y el tiempo de respuesta del sistema durante el control del robot móvil.

Las evaluaciones se realizaron principalmente sobre el robot móvil, debido a que este módulo se encontraba completamente implementado e integrado al momento de la ejecución de las pruebas experimentales. En el caso del robot humanoide bípedo, se realizaron pruebas preliminares de comunicación y activación de acciones mediante ROS, quedando su validación completa planteada como una fase posterior del proyecto.

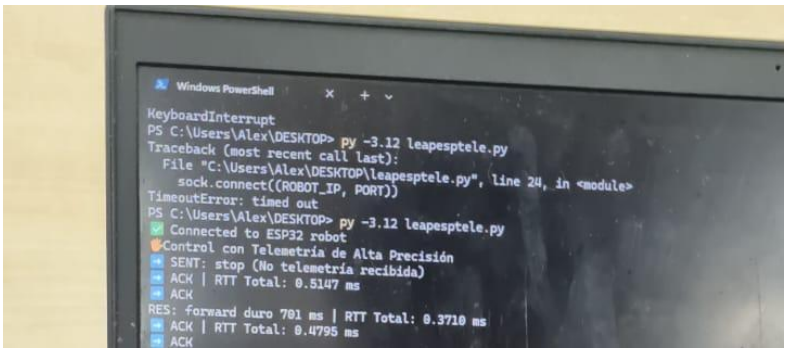
Los resultados presentados incluyen el análisis del tiempo de latencia de comunicación, el comportamiento del sistema ante distintos comandos de movimiento y la estabilidad general del sistema de control gestual.

6.1. Validación del reconocimiento de gestos.

Durante las pruebas experimentales se verifico el correcto funcionamiento del módulo de reconocimiento de gestos implementados mediante el sensor Leap Motion. El sistema fue capaz de identificar de manera consistente los estados de mano abierta y cerrada, así como las combinaciones de gestos definidas para la generación de comandos de movimientos.

Figura 21

Ejemplo de reconocimiento de gesto y generación automática del comando de detención.



Asimismo, se comprobó el adecuado funcionamiento del mecanismo de seguridad implementado mediane el comando de detención por defecto, el cual se activa automáticamente cuando no se detectan ambas manos simultáneamente, cuando la combinación de gestos no corresponde a un comando valido o cuando se presenta perdidas temporales de seguimiento. Este comportamiento permitió evitar desplazamientos no controlados del sistema robótico ante errores de reconocimiento o interrupciones en la señal.

6.2. Evaluación de la comunicación y latencia.

Con el fin de evaluar el desempeño temporal del sistema de teleoperación, se analizó el tiempo de ida y vuelta (round-trip time, RTT) correspondiente al envió de comandos desde el computador hacia el robot móvil y la recepción de las confirmaciones generadas por el sistema robótico.

Los tiempos de comunicación fueron registrados automáticamente mediante el módulo de telemetría implementado en el sistema de control de Python, almacenando los datos en un archivo en formato CSV para su posterior procesamiento y análisis estadísticos. Durante las pruebas se realizaron diez repeticiones de cada comando de movimiento (avance, retroceso, giro a la izquierda y giro a la derecha).

Los datos experimentados obtenidos en el archivo Excel, para el análisis estadístico y la generación de las gráficas se utilizó el entorno Matlab, a partir del cual se calcularon los valores mínimos, máximos y promedio de los tiempos de latencia para cada comando de movimiento, así como su representación gráfica.

En la tabla 3 se presenta los valores mínimo, máximo y promedio del tiempo de latencia obtenida para cada uno de los comandos evaluados.

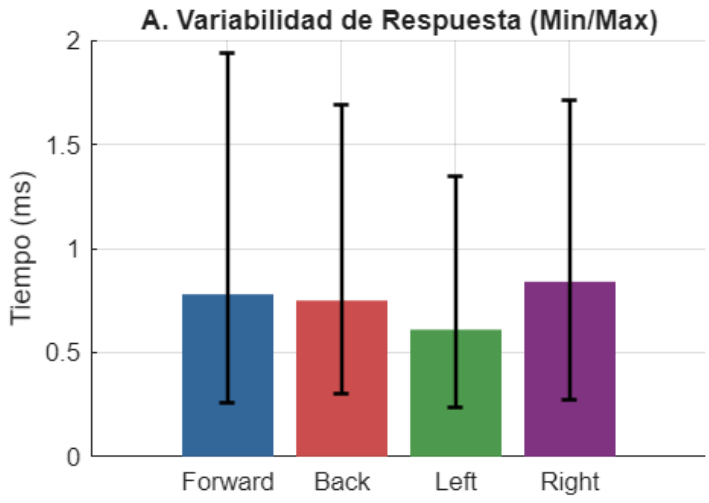
Tabla 3
Tiempos de latencia de comunicación del sistema de teleoperación.

Comando	RTT mínimo (ms)	RTT máximo (ms)	RTT promedio (ms)
forward	0.2878	1.8257	0.7595
back	0.3016	0.9376	0.6053
left	0.2468	1.3559	0.651
right	0.2714	1.4971	0.7189

Los resultados obtenidos muestran que el tiempo de latencia promedio del sistema de comunicación se mantuvo por debajo de 2ms para todos los comandos evaluados, lo que evidencia una respuesta prácticamente inmediata entre el computador y el robot móvil. Este comportamiento permite una interacción fluida entre los movimientos manuales del usuario y la ejecución de los desplazamientos del robot.

Se observa que los distintos comandos presentan tiempos de respuesta similares, lo que indica un comportamiento estable y uniforme del sistema de comunicación inalámbrica. La baja variabilidad de los tiempos registrados demuestra la confiabilidad del enlace TPC implementado y la adecuada sincronización entre el módulo de control en Python y el sistema de control del robot móvil.

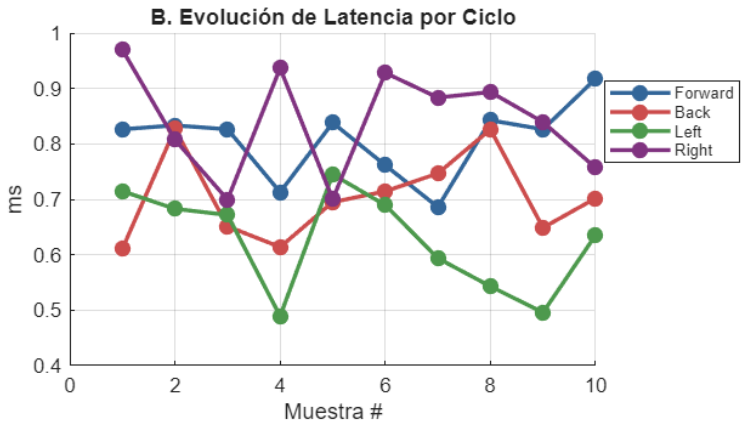
Figura 22
Variabilidad de los tiempos de latencia para los distintos comandos de movimiento.



La figura 18 muestra la variabilidad de los tiempos de latencia obtenidos para los distintos comandos de movimiento. Se observan que los valores promedios se mantienen por debajo de 1ms para todos los comandos evaluados, mientras que los valores máximos no superan los 2ms, lo que evidencia una respuesta rápida del sistema de comunicación.

El comando de giro a la izquierda presento el menor tiempo promedio de respuesta, mientras que el comando de giro a la derecha mostro el mayor valor promedio. No obstante, la diferencias entre comandos son reducidas, lo que indica un comportamiento estable y uniforme del sistema de teleoperación.

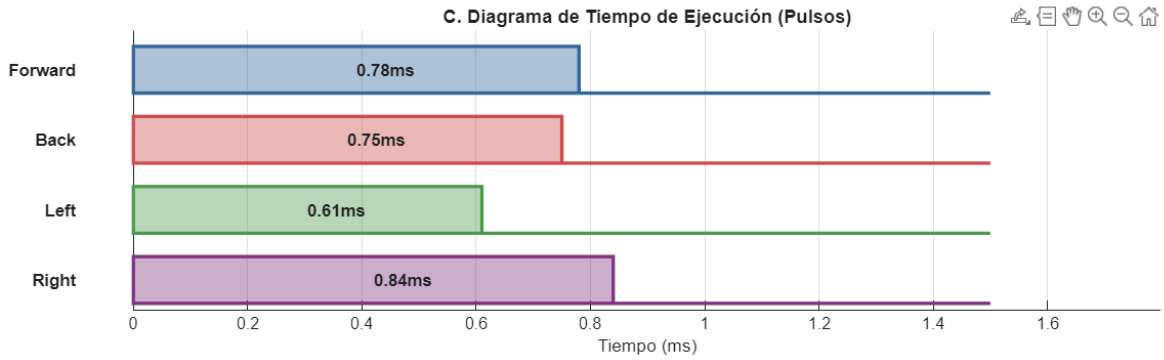
Figura 23
Evolución del tiempo de latencia en función del número de intentos para los comandos de movimiento.



La figura 19 muestra la evolución de latencia registrado en función del número de intentos realizados durante las pruebas experimentales. Se observa que los valores de latencia se mantienen prácticamente constantes a lo largo de las repeticiones, sin presentar incremento abrupto ni comportamientos inestables.

Estos resultados confirman que la arquitectura de comunicación propuesta es adecuada para aplicaciones de teleoperación en tiempo real, permitiendo un control preciso y seguro del sistema robótico mediante gestos manuales.

Figura 24
Diagrama de tiempo de ejecución de los comandos de movimiento.



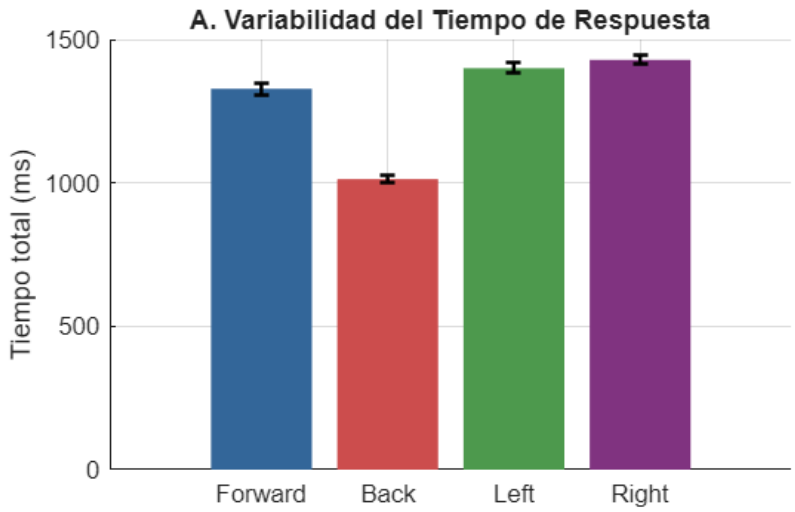
Adicionalmente, se realizó una medición del tiempo total de respuestas del robot humanoide bípedo, considerando el intervalo desde la generación del comando hasta la finalización de la rutina de movimiento ejecutado por el robot. A diferencia del robot móvil, en este caso el tiempo dominante corresponde a la ejecución mecánica de la locomoción del humanoide.

Tabla 4
Tiempos de respuesta del robot humanoide bípedo por los distintos movimientos ejecutados.

Movimiento	Promedio Total(ms)	Mín(ms)	Máx(ms)	Promedio recepción de red(ms)
forward	1328.014	1305.1	1348.12	0.107
back	1012.912	1002.11	1025.11	0.107
left	1400.014	1382.1	1420.13	0.107
right	1429.212	1415.11	1445.11	0.107

En la tabla 4 se muestran los valores expresados en milisegundos e incluyen el tiempo de transmisión por red y la ejecución mecánica del movimiento del robot.

Figura **25**
Variabilidad del tiempo de respuesta del robot humanoide para cada movimiento.

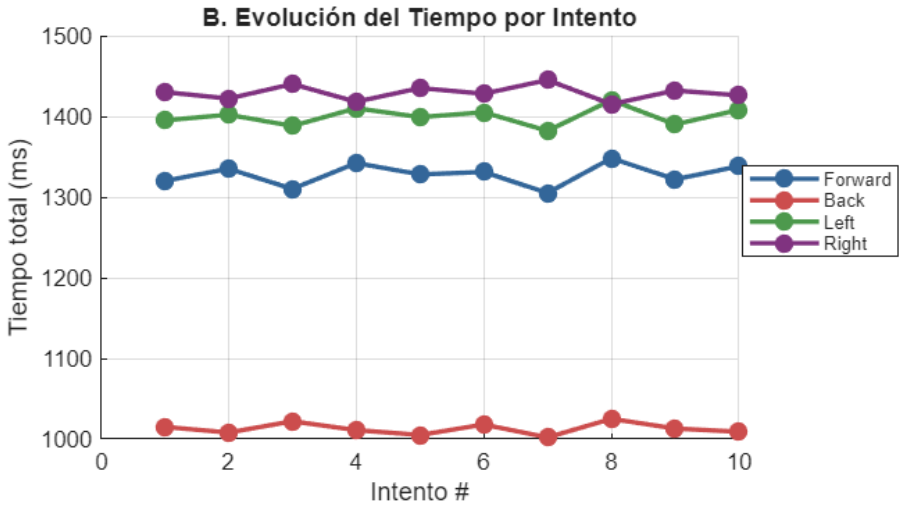


La figura 22 presenta el análisis del tiempo de respuesta del robot humanoide bípedo para los distintos comandos de movimiento. En la gráfica se puede observar la variabilidad estadística (mínimo, y máximo) obtenida a partir de diez repeticiones por cada movimiento.

Los resultados muestran que el comando de retroceso (Back) presenta el menor tiempo promedio de respuesta (1013ms), mientras que los movimientos laterales (Left y Right) registran los mayores tiempos (1400ms y 1429ms respectivamente).

Figura 26

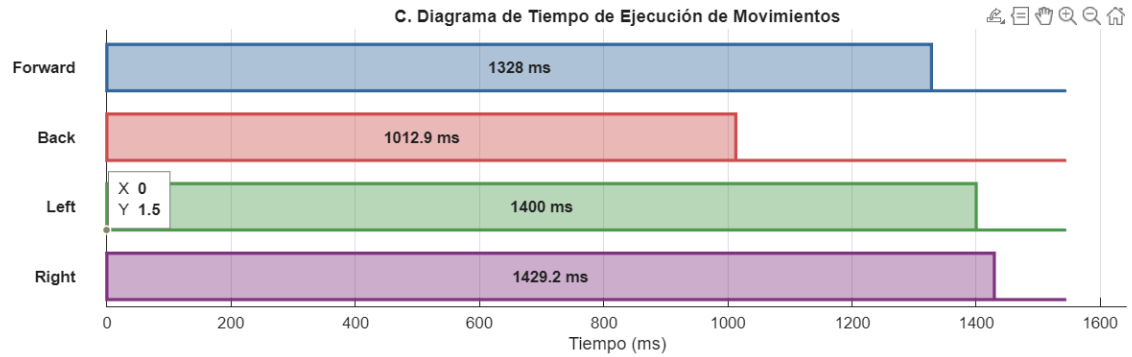
Evolución del tiempo de respuesta del robot humanoide por número de intentos.



La figura 23 muestra la evolución del tiempo de respuesta en función del número de intentos. Se evidencia que las curvas se mantienen prácticamente estables a lo largo de las diez repeticiones, sin incrementos abruptos ni comportamientos erráticos. Esto indica una alta repetibilidad del sistema y confirma que el tiempo de respuesta no depende del instante de ejecución sino de la rutina de movimiento programados en el robot.

Figura 27

Comparación del tiempo de ejecución de los movimientos del robot humanoide.



La figura 24 representa los movimientos como pulsos de duración equivalentes a su tiempo promedio de ejecución. Esta visualización permite comparar directamente cuanto tarda cada acción en completarse. Se aprecia claramente que el movimiento Back finaliza primero, seguido por Forward, mientras que Left y Right son los más largos. Este resultado

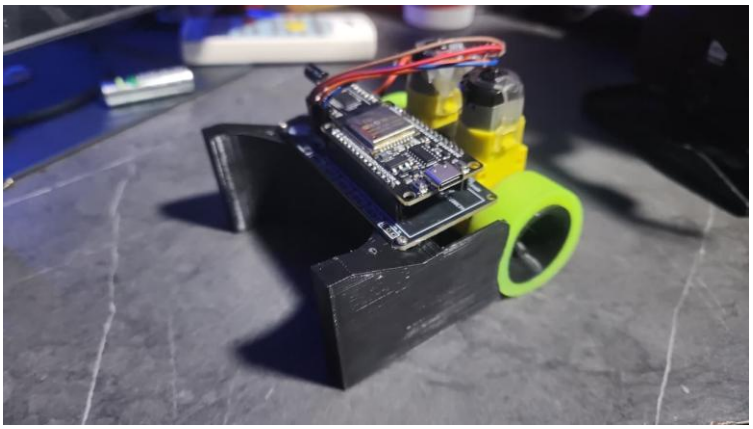
confirma que e retraso dominante en el robot humanoide es de tiempos de captura del gesto y transmisión por red son despreciables frente al tiempo requerido para la locomoción física del robot.

En conjunto, los resultados validan la estabilidad y confiabilidad del sistema de teleoperación aplicado al robot humanoide, demostrando que las variaciones temporales observadas están asociadas principalmente a la dinámica de movimientos del robot y no a fallas en la detención de gestos ni en la comunicación.

6.3. Funcionamiento del sistema con el robot móvil.

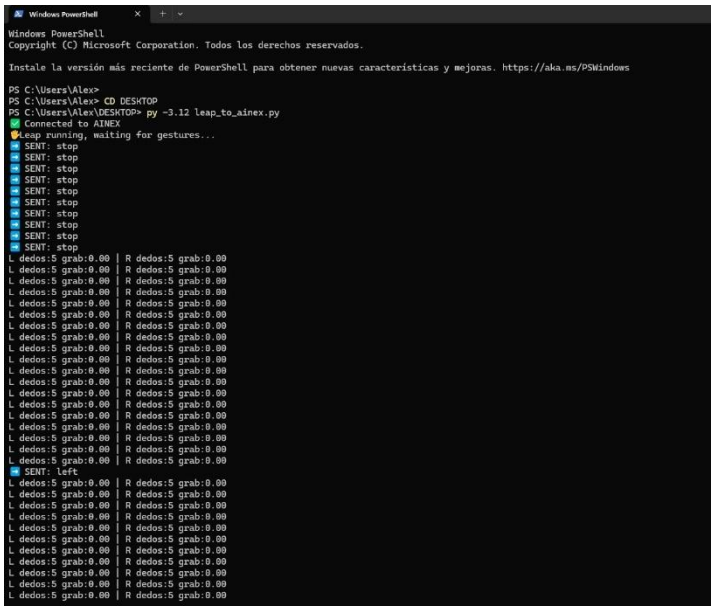
Durante las pruebas realizadas se verifico el correcto funcionamiento del sistema de teleoperación gesticular al robot móvil. El robot fue capaz de ejecutar de manera adecuada los comandos de avance, retroceso y giros laterales generados a partir de los gestos manuales del usuario.

Figura 28
Imagen del robot móvil a utilizar.



El sistema respondió de forma estable ante secuencias continuas de comandos, mantenido un comportamiento controlado y sin presentar movimientos erráticos. Asimismo, el mecanismo de parada automática permitió detener el robot de manera inmediata, ante la ausencia de comandos validos o perdidas temporales de comunicación, garantizando un funcionamiento seguro durante las pruebas experimentales.

Figura 29
Ejecución de comando de movimiento del robot móvil mediante control gestual.



6.4. Pruebas preliminares con el robot humanoide bípedo.

En el caso del robot humanoide bípedo, se realizaron pruebas preliminares orientadas a verificar la correcta recepción de comandos y la activación de acciones de movimientos mediante la plataforma ROS. El nodo implementando fue capaz de recibir instrucciones desde el módulo de control en Python y ejecutar las acciones correspondientes preprogramadas en el sistema del robot.

Figura 30

Activación de acciones de movimientos en el robot humanoide.

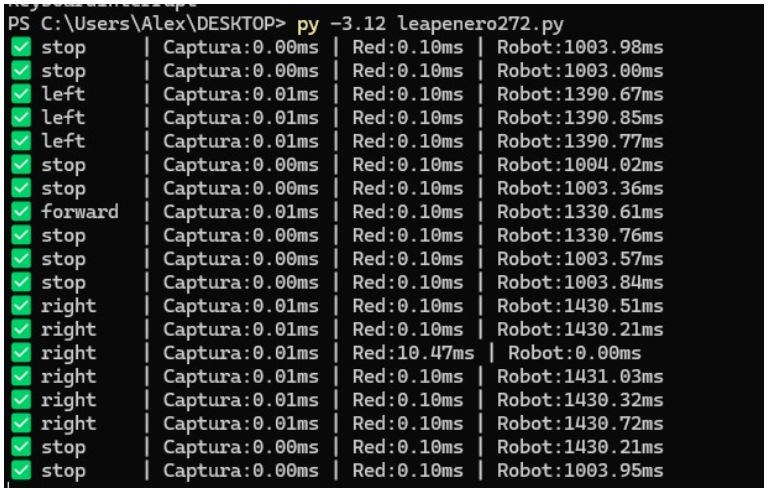


Figura 31
Visualización del robot humanoide a usar para el proyecto.

En este capítulo se presentaron los resultados experimentales obtenidos durante la validación del sistema de teleoperación gesticular. Los análisis realizados permitieron comprobar el adecuado desempeño del reconocimiento de gestos, la confiabilidad de la comunicación inalámbrica y la viabilidad de la arquitectura propuesta para el control robótico en tiempo real.

7. CRONOGRAMA DE ACTIVIDADES

En la Tabla 1 se muestran las actividades a realizar durante los meses de noviembre, diciembre, enero en un período de 4 semanas por cada mes, en la cual se puede visualizar cada uno de los pasos que conlleva realizar el desarrollo.

Tabla 5

Cronograma de actividades

Actividades	***Tiempo de duración***											
	Meses	***Noviembre***				***Diciembre***				***Enero***			
	Semanas	***1***	***2***	***3***	***4***	***1***	***2***	***3***	***4***	***1***	***2***	***3***	***4***
Elección del tema del proyecto de titulación.		X											
Definición de objetivos generales y específicos.			X										
Planteamiento e identificación del problema.				X									
Revisión y selección de las metodologías a utilizar.					X								
Investigación, programaciones y robot móvil durante la etapa 1 y 2						X							
Diseño de la estructura y configuración del robot durante la etapa 1.							X						
Desarrollo de la programación de las rutinas gesticulares para el movimiento del robot durante la etapa 3.								X	X				
Desarrollo del enlace con el robot móvil y el sensor para el movimiento gesticular durante la etapa 4 y 5.								X	X	X			
Implementación en el robot y ajuste de los parámetros del sensor durante a etapa 6.									X	X	X		
Pruebas finales de todo movimiento gesticular durante la etapa 7.											X	X	

Nota. En esta tabla se desglosa de manera semanal cada actividad que se va a llevar a cabo durante los meses mencionados.

8. PRESUPUESTO

En la Tabla 2 se da a conocer el presupuesto sobre las horas de ingeniería que se tomó en cuenta al momento de desarrollar el proyecto.

Tabla 6
Cuadro de presupuesto

DETALLE	*CANTIDAD*	*COSTO UNITARIO*	*COSTO TOTAL*
Horas de ingeniería	***80h***	***$3.00***	***$240.00***
Sensor Leap Motion	***1***	***$120.00***	***$120.00***
**Robot humanoide*	***1***	***$609.99***	***$609.99***
**Robot móvil*	***1***	***$1,039.99***	***$1,039.99***
TOTAL			***$2009.98***

Nota. Los * son elementos de propiedad de los autores.

9. CONCLUSIONES

- Se implementó el control de movimiento de un robot móvil utilizando el sensor Leap Motion, logrando el concepto general del proyecto y demostrando que el reconocimiento de gestos permitió una teleoperación más eficiente, intuitiva y en tiempo real.
- Se diseñó y configuró la estructura electrónica del robot móvil para integrar correctamente los componentes, asegurando su estabilidad en operación, que acepte los comandos de gestos del usuario de manera adecuada o a tiempo, y que opere como se espera, dependiendo de la dirección del gesto del usuario.
- La interfaz de comunicación establecida por el sensor Leap Motion y el sistema de control del robot ha logrado una alta tasa de transmisión para los gestos, lo que permite una buena respuesta en tiempo tanto en las líneas de respuesta del robot como del Leap Motion.
- La programación de rutinas de movimiento basadas en gestos de la mano permite interpretar correctamente varios movimientos de la mano convirtiéndolos en la tarea específica que el robot debe realizar para avanzar, retroceder y girar.
- El trabajo del sistema mostró: el comando por gestos es una alternativa efectiva al sistema de control tradicional, el sistema innovador es un medio potencial para el desarrollo educativo, experimental o industrial futuro.

10. RECOMENDACIONES

- Se recomienda mejorar los algoritmos de reconocimiento de gestos filtrando/suavizando las señales, para minimizar el ruido de los movimientos espontáneos y mejorar la capacidad de respuesta del robot.
- Para el futuro, se propone un codificador o sensor de proximidad en el robot móvil para proporcionar retroalimentación y mejorar la seguridad y precisión de los movimientos.
- El control por gestos para otras subclases de robots, es decir, manipuladores, robots industriales, etc., podría ser utilizado y el rendimiento debería determinarse en casos más complejos y en un contexto práctico.
- Este tipo de proyecto debería incluirse en los laboratorios académicos, porque fomenta el aprendizaje práctico de los estudiantes en robótica, automatización e interacción humano-robot.
- Por último, se sugiere potenciar la investigación de la interfaz de gestos como herramientas de accesibilidad y desarrollar soluciones inclusivas que permitan a las personas con discapacidades motoras participar en actividades técnicas y profesionales.

11. REFERENCIAS

Rodríguez Baeza, E., Vergara Betanc, A., & Osorio Verde, I. (13 de enero de 2022). *pistas educativas*. Obtenido de Encabezado de página: https://pistaseducativas.celaya.tecnm.mx/index.php/pistas/article/view/2791

AER Automation. (22 de julio de 2024). *Robótica móvil: Características, beneficios y casos de éxito*. Obtenido de https://www.automaticaeinstrumentacion.com/texto-diario/mostrar/4940677/robotica-movil

AS, R. (28 de Julio de 2022). *7universum*. Obtenido de PERSPECTIVAS DE EMPLEO PARA LAS PERSONAS CON DISCAPACIDAD EN ECUADOR: https://7universum.com/ru/social/archive/item/14111

Bachmann, D., Weichert, F., & Rinkernauer, G. (15 de Noviembre de 2015). *MDPI*. Obtenido de Evaluation of the Leap Motion Controller as a New Contact-Free Pointing Device: https://www.mdpi.com/1424-8220/15/1/214?utm

Bermúdez, G. (22 de Marzo de 2022). *ROBOTS MÓVILES.* Obtenido de https://files01.core.ac.uk/download/pdf/229164924.pdf

Bird, J. J. (22 de Febrero de 2022). *Cornell University*. Obtenido de Características estadísticas y espaciotemporales de los gestos de las manos para el reconocimiento del lenguaje de señas mediante el sensor Leap Motion: https://arxiv.org/abs/2202.11005?

Canio, & Di, D. (2019). *The Impact of Robot Gestures on Student Reasoning About Geometrical Conjectures.* Torino: Bachelor's Degree, Politecnico di Torino, Turin, Italy, 2019.

CONADIS. (18 de Agosto de 2025). *Consejo Nacional para la Igualdad de Discapacidades*. Obtenido de https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

Cortes Rogriguez, J. A., Sandoval, F. A., & Trujillo Pascual, I. R. (02 de Diciembre de 2025). *IMPLEMENTACIÓN DE UN CONTROL GESTUAL*. Obtenido de https://d1wqtxts1xzle7.cloudfront.net/60466916/IMPLEMENTACION_DE_UN_CONTROL_GESTUAL_EN_MOVIL_INALAMBRICO20190902-80649-t6hxuh-libre.pdf?1567448572=&response-content-disposition=inline%3B+filename%3DINSTITUTO_POLITECNICO_NACIONAL_ESCUELA_S.pdf&Expires=1769

Dr. Bradley , & S. Duerstock,. (24 de junio de 2020). *Report on the Use of Assistive Robotics to Aid Persons with*. Obtenido de https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1003&context=ugcw&utm_source=

Friendly Captcha. (23 de Diciembre de 2023). *¿Qué es la accesibilidad (en tecnología)?* Obtenido de https://friendlycaptcha.com/es/wiki/what-is-accessibility-in-technology/

Guachamín, Z., & Alejandro, J. (16 de Febrero de 2018). *Implementación de un sistema de clasificación de gestos del brazo humano utilizando Myo Armband para mando a distancia de un brazo robótico de 3GDL*. Obtenido de https://bibdigital.epn.edu.ec/handle/15000/19190

Guna, J., Jakus, G., & Sodnik, J. (21 de Febrero de 2021). *MDPI*. Obtenido de Leap Motion Sensor and Its Suitability: https://www.mdpi.com/1424-8220/14/2/3702?utm

Guna, J., Jakus, G., Pogacnik, M., Tomazic, S., & Sodink, J. (21 de Febrero de 2015). *PMC PubMed Central*. Obtenido de An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking: https://pmc.ncbi.nlm.nih.gov/articles/PMC3958287/?ut

IFR. (18 de Noviembre de 2023). *International Federation of Robotics* . Obtenido de World Robotics Report 2023: https://ifr.org/

Kajan, S., Goja, J., Matejicka, P., Minar, M., Pavlovicova, J., & Kosutzka, Z. (4 de Agosto de 2024). *Electrical Engineering*. Obtenido de Real-time visual verification of leap motion controller measurements for reliable finger tapping test in Parkinson's disease: https://reference-global.com/article/10.2478/jee-2024-0039?utm

Koenig, A., Rodrigez, F., & Secoli, R. (15 de Marzo de 2020). *Gesture-Based Teleoperated Grasping for Educational Robotics*. Obtenido de https://axkoenig.com/downloads/research/roman_paper.pdf?utm_source=

KYLEDESING. (20 de Enero de 2025). *Leap Motion - Gesture Controller*. Obtenido de https://kylehay.com/leap-motion-controller

LEAP MOTION ULTRALEAP. (23 de Diciembre de 2021). *ULTRALEAP*. Obtenido de What are the specifications of the Leap Motion Controller?: https://www.leapmotiontechnology.com/support-sub/hc/en-us/articles/360004476658-What-are-the-specifications-of-the-Leap-Motion-Controller-?utm

López, J. T. (23 de Diciembre de 2021). *Universidad Zaragoza*. Obtenido de Reconocimiento e interpretación de gesto con dispositivo leap motion: https://zaguan.unizar.es/record/12916/files/TAZ-TFM-2013-1096.pdf

Mariuxi, V., Intriago, E., & García, M. (25 de Julio de 2022). *La robótica en el ámbito educativo de Ecuador*. Obtenido de Dialnet: https://dialnet.unirioja.es/servlet/articulo?codigo=8955512

Moreira, P., Neto, P., & Pires, N. (9 de Septiembre de 2013). *Corenll University*. Obtenido de High-level programming and control for industrial robotics: using a hand-held

accelerometer-based input device for gesture and posture recognition: https://arxiv.org/abs/1309.2093?utm_source=

Parreño Alvarez, M. (22 de Marzo de 2021). *Desarrollo de un juego didáctico mediante sensores leap motion para estimular el aprendizaje del lenguaje básico de señas ecuatoriano.* Obtenido de UPS QUITO: https://dspace.ups.edu.ec/bitstream/123456789/15289/1/UPS%20-%20ST003471.pdf

Pastor Bernal, D. (21 de Julio de 2023). *Teleoperación de un robot para acciones de manipulación con dispositivos y sensores de bajo coste.* Obtenido de Escuela Politécnica Superior: https://rua.ua.es/server/api/core/bitstreams/3c6999ac-f1d9-4d71-b0a9-201fa36f57a2/content

PC MAG. (22 de enero de 2024). *Leap Motion Controller*. Obtenido de https://au.pcmag.com/migrated-39864-input-devices/5086/leap-motion-controller-review

RobotShop. (22 de Abril de 2024). *RobotS*. Obtenido de controlador leap motion: https://www.robotshop.com/es/products/controlador-leap-motion?utm

Shukla, D. (9 de julio de 2020). *Leap Motion Controller*. Obtenido de https://www.electronicsforu.com/india-corner/innovations-innovators/leap-motion-controller

TME. (12 de septiembre de 2021). *Sensor leap motion*. Obtenido de https://tiendadeelectronica.mx/tienda/leap-motion-controller/?srsltid=AfmBOoqbqPy4gdXETrJhaAF_yQQqqc3nUqeCOMyc67qEnYOC4CBgo5vU

ULTRALEAP. (24 de Agosto de 2025). *Leap Motion*. Obtenido de Ultraleap Hand Tracking Overview: https://docs.ultraleap.com/hand-tracking/index.html

ULTRALEAP. (19 de Enero de 2026). *Leap Concepts*. Obtenido de https://docs.ultraleap.com/api-reference/tracking-api/leapc-guide/leap-concepts.html

Universidad de arte Madri. (12 de Junio de 2023). *¿Qué es el motion caprute?* Obtenido de https://taiarts.com/blog/motion-capture/

Vallejo, R. (13 de Abril de 2023). *Este método de enseñanza fomenta el pensamiento computacional que prepara a las nuevas generaciones para el mercado laboral*. Obtenido de Robótica educativa: qué es, tipos y ventajas: https://www.telefonica.com/es/sala-comunicacion/blog/robotica-educativa-tipos-y-ventajas/

WIT. (18 de Noviembre de 2023). *WIT AUTOMATIZACIÓN*. Obtenido de Métodos y Lenguajes de programación en la Robótica industrial:

https://witautomatizacion.es/metodos-y-lenguajes-de-programacion-en-robotica-industrial/

12. ANEXOS

En esta parte se presentan los anexos correspondientes al desarrollo del proyecto, los cuales contienen el material complementario que respalda la implementación del sistema propuesto. Se incluye los códigos fuentes de los diferentes módulos desarrollados, así como archivos y configuraciones relevantes utilizados durante la ejecución de las pruebas experimentales.

Figura 32
Código fuente del módulo de reconocimiento gestual y generación de comandos en Python para el control humanoide.

```python
import socket
import time
import leap

ROBOT_IP = "192.168.149.1"
PORT = 5005

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((ROBOT_IP, PORT))
print("Connected to AINEX")

OPEN_TH = 0.5
CLOSED_TH = 0.7
COOLDOWN = 0.7

last_cmd = ""
last_time = 0


def send_cmd(cmd):  7 usages
    global last_cmd, last_time
    now = time.time()

    if cmd == last_cmd:
        return
    if now - last_time < COOLDOWN:
        return

    sock.sendall((cmd + "\n").encode())
    print("➡ SENT:", cmd)

```

```
        last_cmd = cmd
        last_time = now


class LeapListener(leap.Listener):  1 usage
    def on_tracking_event(self, event):
        hands = event.hands

        if len(hands) < 2:
            send_cmd("stop")
            return

        left = None
        right = None

        for h in hands:
            if h.type == leap.HandType.Left:
                left = h
            elif h.type == leap.HandType.Right:
                right = h

        if not left or not right:
            send_cmd("stop")
            return

        lg = left.grab_strength
        rg = right.grab_strength

        print(f"L:{lg:.2f}  R:{rg:.2f}")

        left_open = lg < OPEN_TH
        right_open = rg < OPEN_TH
        left_closed = lg > CLOSED_TH
        right_closed = rg > CLOSED_TH

        if left_open and right_closed:
            send_cmd("forward")
        elif left_closed and right_open:
            send_cmd("back")
        elif left_closed and right_closed:
            send_cmd("right")
        elif left_open and right_open:
            send_cmd("left")
        else:
            send_cmd("stop")
```

```
77
78
79  def main():  1 usage
80      connection = leap.Connection()
81      listener = LeapListener()
82      connection.add_listener(listener)
83
84      print("🖐 Leap running, waiting for gestures...")
85
86      with connection.open():
87          while True:
88              time.sleep(1)
89
90
91  if __name__ == "__main__":
92      main()
93
```

Figura 33
Código fuente del módulo cliente TCP y registro de telemetría.

```
1   import socket                                                    ✓ 1
2
3   ROBOT_IP = "192.168.149.1"  # IP del AINEX
4   PORT = 5005
5
6   print("Connecting to robot...")
7
8   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9   s.connect((ROBOT_IP, PORT))
10
11  print("Connected to robot")
12
13  s.sendall(b"HELLO_FROM_PC")
14
15  data = s.recv(1024).decode()
16  print("Robot says:", data)
17
18  s.close()
19  print("Connection closed")
```

Figura 34

Código fuente del servidor TCP y control del robot móvil implementado en ESP32.

```
#include <WiFi.h>

const char* ssid = "TESISGOD";
const char* password = "ROBOTMOVIL";

WiFiServer server(6005);
WiFiClient client;

// ====== PINES ======
#define MOTOR_IZQ_EN 25
#define MOTOR_IZQ_IN1 26
#define MOTOR_IZQ_IN2 27
#define MOTOR_DER_EN 13
#define MOTOR_DER_IN1 12
#define MOTOR_DER_IN2 14

#define SPEED 110
#define TURN_SPEED 85

String lastCmd = "stop";
unsigned long startTime = 0;
unsigned long lastCmdTime = 0;
const unsigned long TIMEOUT = 700;

// --- FUNCIONES DE MOVIMIENTO ---

void stopMotors() {
  if (lastCmd != "stop") {
    unsigned long duration = millis() - startTime;
    Serial.print("Movimiento_ms:"); Serial.println(duration);
    if(client.connected()) {
      client.printf("RES: %s duro %d ms\n", lastCmd.c_str(), (int)duration);
    }
  }
  // Frenado (Poner ambos en HIGH bloquea el motor)
  digitalWrite(MOTOR_IZQ_IN1, HIGH);
  digitalWrite(MOTOR_IZQ_IN2, HIGH);
  digitalWrite(MOTOR_DER_IN1, HIGH);
  digitalWrite(MOTOR_DER_IN2, HIGH);
  analogWrite(MOTOR_IZQ_EN, 0);
  analogWrite(MOTOR_DER_EN, 0);
}

void moveForward() {
  digitalWrite(MOTOR_IZQ_IN1, LOW); digitalWrite(MOTOR_IZQ_IN2, HIGH);
  digitalWrite(MOTOR_DER_IN1, HIGH); digitalWrite(MOTOR_DER_IN2, LOW);
  analogWrite(MOTOR_IZQ_EN, SPEED); analogWrite(MOTOR_DER_EN, SPEED);
}
```

```
49
50  void moveBackward() {
51    digitalWrite(MOTOR_IZQ_IN1, HIGH); digitalWrite(MOTOR_IZQ_IN2, LOW);
52    digitalWrite(MOTOR_DER_IN1, LOW); digitalWrite(MOTOR_DER_IN2, HIGH);
53    analogWrite(MOTOR_IZQ_EN, SPEED); analogWrite(MOTOR_DER_EN, SPEED);
54  }
55
56  void turnRight() {
57    // Rueda derecha adelante, rueda izquierda atrás (Giro sobre eje)
58    // Rueda izquierda adelante, rueda derecha atrás (Giro sobre eje)
59    digitalWrite(MOTOR_IZQ_IN1, HIGH); digitalWrite(MOTOR_IZQ_IN2, LOW);
60    digitalWrite(MOTOR_DER_IN1, HIGH); digitalWrite(MOTOR_DER_IN2, LOW);
61    analogWrite(MOTOR_IZQ_EN, TURN_SPEED); analogWrite(MOTOR_DER_EN, TURN_SPEED);
62  }
63
64  void turnLeft() {
65    // Rueda derecha adelante, rueda izquierda atrás (Giro sobre eje)
66    digitalWrite(MOTOR_IZQ_IN1, LOW); digitalWrite(MOTOR_IZQ_IN2, HIGH);
67    digitalWrite(MOTOR_DER_IN1, LOW); digitalWrite(MOTOR_DER_IN2, HIGH);
68    analogWrite(MOTOR_IZQ_EN, TURN_SPEED); analogWrite(MOTOR_DER_EN, TURN_SPEED);
69  }
70
71  // --- PROCESAMIENTO DE COMANDOS ---
72
73  void executeCommand(String cmd) {
74    unsigned long receiveTime = millis();
75    cmd.trim();
76    if (cmd == "") return;
77
78    // Detener motores antes de cambiar de dirección si ya se estaba moviendo
79    if (lastCmd != "stop" && cmd != lastCmd) {
80      // No llamamos a stopMotors aquí para evitar spam de mensajes,
81      // solo reseteamos pines si es necesario.
82    }
83
84    lastCmd = cmd;
85    lastCmdTime = receiveTime;
86    startTime = receiveTime;
87
88    if (cmd == "forward") moveForward();
89    else if (cmd == "back") moveBackward();
90    else if (cmd == "left") turnLeft();
91    else if (cmd == "right") turnRight();
92    else if (cmd == "stop") stopMotors();
93
94    if (client.connected()) {
95      client.println("ACK");
96    }
97  }
```

```
98
99  // --- SETUP Y LOOP ---
100
101 void setup() {
102   Serial.begin(115200);
103
104   pinMode(MOTOR_IZQ_EN, OUTPUT);
105   pinMode(MOTOR_IZQ_IN1, OUTPUT);
106   pinMode(MOTOR_IZQ_IN2, OUTPUT);
107   pinMode(MOTOR_DER_EN, OUTPUT);
108   pinMode(MOTOR_DER_IN1, OUTPUT);
109   pinMode(MOTOR_DER_IN2, OUTPUT);
110
111   stopMotors();
112
113   WiFi.begin(ssid, password);
114   Serial.print("Conectando a WiFi");
115   while (WiFi.status() != WL_CONNECTED) {
116     delay(500);
117     Serial.print(".");
118   }
119   Serial.println("\nWiFi Conectado");
120   Serial.print("IP: "); Serial.println(WiFi.localIP());
121
122   server.begin();
123 }
124
```

```
125 void loop() {
126   // Gestionar conexión de cliente
127   if (!client || !client.connected()) {
128     client = server.available();
129   } else {
130     if (client.available()) {
131       String cmd = client.readStringUntil('\n');
132       executeCommand(cmd);
133     }
134   }
135
136   // Sistema de seguridad: si no recibe comandos, se detiene
137   if (millis() - lastCmdTime > TIMEOUT && lastCmd != "stop") {
138     stopMotors();
139     lastCmd = "stop";
140   }
141 }
```

Figura 35
Código fuente del nodo ROS para la recepción de comandos y activación de acciones del robot humanoide.

```python
#!/usr/bin/env python3
# coding: utf-8

import socket
import rospy
from ainex_example.color_common import Common

HOST = "0.0.0.0"   # Escucha cualquier IP
PORT = 5005        # Puerto del socket

class RobotActionServer(Common):  1 usage
    def __init__(self):
        print("=== ROBOT ACTION SERVER STARTING ===")

        rospy.init_node("robot_action_server")

        # Inicializa AINEX (motores, acciones, etc)
        super().__init__("robot_action_server", 500, 260)

        print("AINEX initialized correctly")

        # Socket TCP
        self.server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, value: 1)
        self.server.bind((HOST, PORT))
        self.server.listen(1)

        print(f"?? Waiting for PC connection on port {PORT} ...")

        self.client, addr = self.server.accept()
        print(f"? PC connected from {addr}")

        self.run()

    def execute_cmd(self, cmd):  1 usage
        try:
            cmd = cmd.strip().lower()

            if cmd == "forward":
                print("FORWARD")
                self.motion_manager.run_action("walk_ready")
                rospy.sleep(0.3)
                self.motion_manager.run_action("forward")
```

```
44
45              elif cmd == "back":
46                  print("BACK")
47                  self.motion_manager.run_action("walk_ready")
48                  rospy.sleep(0.3)
49                  self.motion_manager.run_action("back")
50
51              elif cmd == "left":
52                  print("TURN LEFT")
53                  self.motion_manager.run_action("walk_ready")
54                  rospy.sleep(0.3)
55                  self.motion_manager.run_action("turn_left")
56
57              elif cmd == "right":
58                  print("TURN RIGHT")
59                  self.motion_manager.run_action("walk_ready")
60                  rospy.sleep(0.3)
61                  self.motion_manager.run_action("turn_right")
62
63              elif cmd == "stop":
64                  print("STOP")
65                  self.motion_manager.run_action("stand")
66
```

```
67              # ==========================
68              # NUEVOS MOVIMIENTOS
69              # ==========================
70
71              elif cmd == "greet":
72                  print("GREET")
73                  self.motion_manager.run_action("walk_ready")
74                  rospy.sleep(0.3)
75                  self.motion_manager.run_action("greet")
76
77              elif cmd == "bye":
78                  print("DESPEDIDA")
79                  self.motion_manager.run_action("walk_ready")
80                  rospy.sleep(0.3)
81                  self.motion_manager.run_action("despedida")
82
83              else:
84                  print(f"? Unknown command: {cmd}")
85
86          except Exception as e:
87              print("? Error executing command:", e)
88
```

```python
    def run(self):  1 usage
        try:
            while not rospy.is_shutdown():
                data = self.client.recv(1024)
                if not data:
                    print("PC disconnected")
                    break

                cmd = data.decode("utf-8")
                print(f"?? Command received: {cmd}")
                self.execute_cmd(cmd)

        except KeyboardInterrupt:
            pass

        finally:
            self.client.close()
            self.server.close()
            self.motion_manager.run_action("stand")
            print("?? Robot action server stopped")


if __name__ == "__main__":
    RobotActionServer()
```

Figura 36
Código fuente de prueba del cliente TCP para control manual por teclado y validación de la comunicación

```python
import socket

ROBOT_IP = "192.168.149.1"
PORT = 5005

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((ROBOT_IP, PORT))

print("Connected. Type commands (forward/back/left/right/stop/exit)")

while True:
    cmd = input(">> ")
    s.sendall(cmd.encode())

    if cmd == "exit":
        break

s.close()
print("Disconnected")
```

Figura 37
Código fuente del control del robot móvil.

```python
import socket
import time
import leap
import csv

# ==========================
# CONFIGURACIÓN
# ==========================
ROBOT_IP = "10.130.218.211"
PORT = 6005

COOLDOWN = 0.2
last_time = 0
last_cmd = ""

# Crear archivo y escribir cabecera
with open("log_telemetria.csv", "a", newline="") as f:
    writer = csv.writer(f)
    if f.tell() == 0:
        writer.writerow(["Timestamp", "Comando", "Respuesta_Robot", "RTT_ms"])

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.settimeout(0.1)
sock.connect((ROBOT_IP, PORT))
print("Connected to ESP32 robot")
```

```python
def send_cmd(cmd):  7 usages
    global last_time, last_cmd
    # Usamos perf_counter para el cooldown también
    now = time.perf_counter()

    if cmd == last_cmd:
        return
    if now - last_time < COOLDOWN:
        return

    try:
        # --- ALTA PRECISIÓN ---
        start_t = time.perf_counter()
        sock.sendall((cmd + "\n").encode())

        response = sock.recv(1024).decode().strip()
        end_t = time.perf_counter()
```

```
            # Cálculo en milisegundos con decimales reales
            rtt_ms = (end_t - start_t) * 1000

            # Ahora verás valores como 0.98, 1.45, etc.
            print(f" {response} | RTT Total: {rtt_ms:.4f} ms")

            with open("log_telemetria.csv", "a", newline="") as f:
                writer = csv.writer(f)
                writer.writerow([time.strftime("%H:%M:%S"), cmd, response, f"{rtt_ms:.4

        except socket.timeout:
            print(f" SENT: {cmd} (No telemetría recibida)")
        except Exception as e:
            print(f" Error: {e}")

        last_cmd = cmd
        last_time = now


class LeapListener(leap.Listener):  1 usage
    def on_tracking_event(self, event):
        hands = event.hands
        if len(hands) < 2:
            send_cmd("stop")
            return
```

```

            left = right = None
            for h in hands:
                if h.type == leap.HandType.Left:
                    left = h
                elif h.type == leap.HandType.Right:
                    right = h

            if not left or not right:
                send_cmd("stop")
                return

            lg, rg = left.grab_strength, right.grab_strength

            l_open, r_open = lg < 0.4, rg < 0.4
            l_closed, r_closed = lg > 0.8, rg > 0.8

            if l_open and r_open:
                send_cmd("forward")
            elif l_closed and r_closed:
                send_cmd("back")
            elif l_open and r_closed:
                send_cmd("left")
```

```
        elif l_closed and r_open:
            send_cmd("right")
        else:
            send_cmd("stop")


def main():  1 usage
    connection = leap.Connection()
    listener = LeapListener()
    connection.add_listener(listener)
    print("Control con Telemetría de Alta Precisión")
    with connection.open():
        while True:
            time.sleep(0.01)


if _name_ == "_main_":
    main()
```

Figura 38
Código test para la altura de las manos.

```
import leap                                                    1 2 27
import time

# ========= PARÁMETROS =========
OPEN_THRESHOLD = 0.7     # mano abierta
CLOSE_THRESHOLD = 0.3    # mano cerrada

HEIGHT_ERGUIDO = 220     # manos claramente arriba
HEIGHT_ENCOGIDO = 100    # manos claramente abajo

PRINT_DELAY = 0.3        # evita spam

# ==============================

class GestureController(leap.Listener):  1 usage
    def __init__(self):
        self.current_direction = None
        self.current_height = "ERGUIDO"
        self.last_print = time.time()
```

```
20
21      # ---------- UTILIDADES ----------
22      def hand_open_ratio(self, hand):  1 usage
23          extended = sum(1 for d in hand.digits if d.is_extended)
24          return extended / 5.0
25
26      def classify_hand(self, hand):  2 usages
27          ratio = self.hand_open_ratio(hand)
28          if ratio >= OPEN_THRESHOLD:
29              return "OPEN"
30          elif ratio <= CLOSE_THRESHOLD:
31              return "CLOSED"
32          return "UNKNOWN"
33
34      # ---------- ALTURA ----------
35      def update_height(self, left, right):  1 usage
36          avg_y = (left.palm.position.y + right.palm.position.y) / 2
37
38          if self.current_height == "ENCOGIDO":
39              if avg_y > HEIGHT_ERGUIDO:
40                  self.current_height = "ERGUIDO"
41                  print(" ALTURA → ERGUIDO")
42
43          elif self.current_height == "ERGUIDO":
44              if avg_y < HEIGHT_ENCOGIDO:
45                  self.current_height = "ENCOGIDO"
46                  print(" ALTURA → ENCOGIDO")
47
48      # ---------- DIRECCIÓN ----------
49      def detect_direction(self, left_state, right_state):  1 usage
50          if left_state == "OPEN" and right_state == "CLOSED":
51              return "ADELANTE"
52          if left_state == "CLOSED" and right_state == "OPEN":
53              return "ATRÁS"
54          if left_state == "CLOSED" and right_state == "CLOSED":
55              return "DERECHA"
56          if left_state == "OPEN" and right_state == "OPEN":
57              return "IZQUIERDA"
58          return None
59
60      # ---------- EVENTO PRINCIPAL ----------
61      def on_tracking_event(self, event):
62          if len(event.hands) < 2:
63              return
64
65          left = next((h for h in event.hands if h.type == leap.HandType.Left), None)
66          right = next((h for h in event.hands if h.type == leap.HandType.Right), None)
67
68          if not left or not right:
69              return
```

```
70
71          left_state = self.classify_hand(left)
72          right_state = self.classify_hand(right)
73
74          self.update_height(left, right)
75          direction = self.detect_direction(left_state, right_state)
76
77          now = time.time()
78          if direction and direction != self.current_direction and now - self.last_print :
79              self.current_direction = direction
80              self.last_print = now
81
82              print(f" MOVIMIENTO → {direction} |  {self.current_height}")
83
84  # ---------- MAIN ----------
85  def main():  1 usage
86      print(" Leap Motion - Control Humanoide PRO")
87      print("Ctrl+C para salir\n")
88
89      listener = GestureController()
90      connection = leap.Connection()
91      connection.add_listener(listener)
92
93      with connection.open():
94          while True:
95              time.sleep(0.01)
96
97  if __name__ == "__main__":
98      main()
```

Figura 39
Código test para gestos con dos manos.

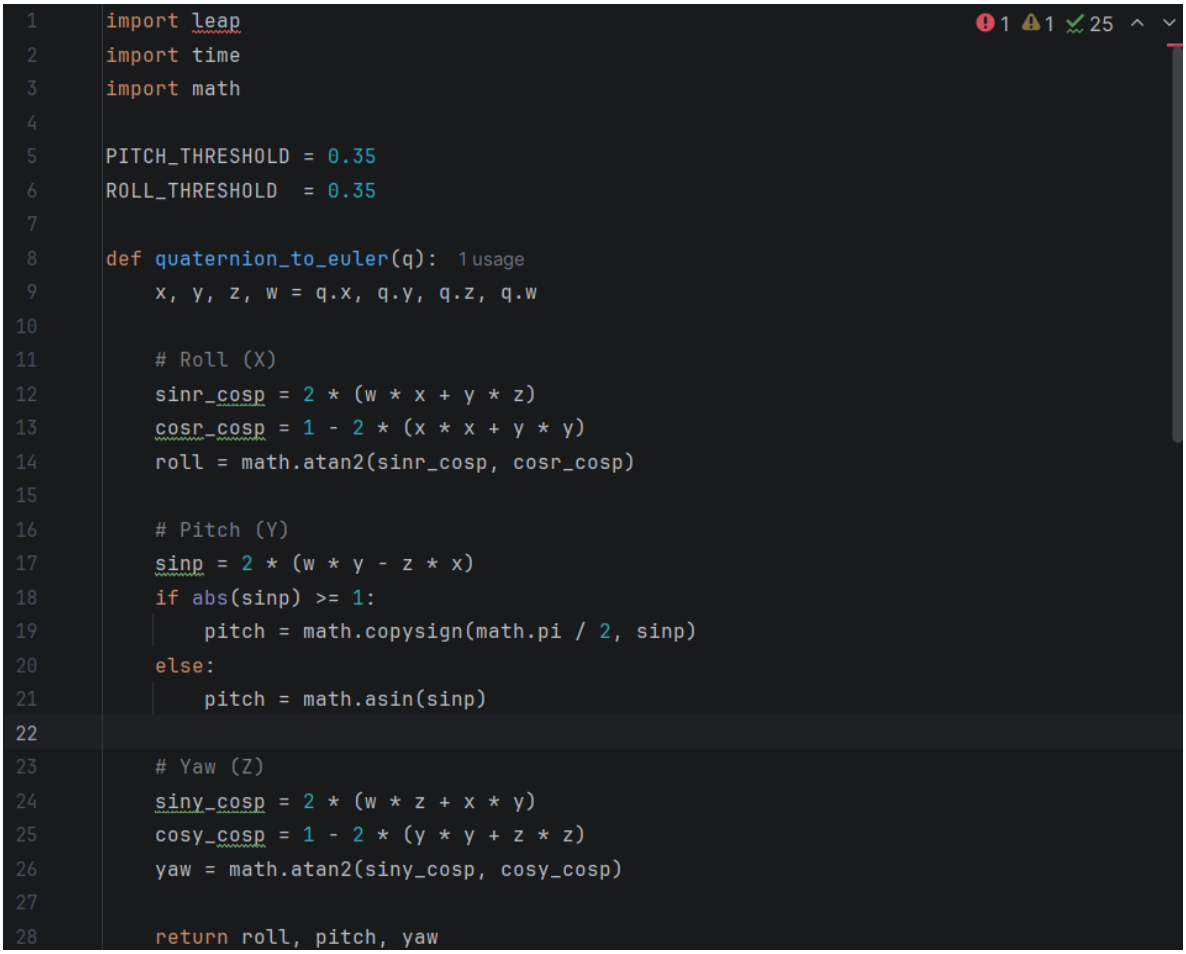
```
1   import leap                                                    1 1 8
2   import time
3
4   OPEN_THRESHOLD = 0.25
5   CLOSED_THRESHOLD = 0.8
6
7   def hand_state(hand):  2 usages
8       if hand.grab_strength < OPEN_THRESHOLD:
9           return "OPEN"
10      elif hand.grab_strength > CLOSED_THRESHOLD:
11          return "CLOSED"
12      else:
13          return "UNKNOWN"
14
15
16  class TwoHandGestureListener(leap.Listener):  1 usage
```

```
    def on_tracking_event(self, event):

        if len(event.hands) < 2:
            return

        left = None
        right = None

        for hand in event.hands:
            if hand.type == leap.HandType.Left:
                left = hand
            elif hand.type == leap.HandType.Right:
                right = hand

        if not left or not right:
            return

        left_state = hand_state(left)
        right_state = hand_state(right)

        if "UNKNOWN" in (left_state, right_state):
            return

        # Patrones de control
        if left_state == "OPEN" and right_state == "CLOSED":
            action = " ADELANTE"

        elif left_state == "CLOSED" and right_state == "OPEN":
            action = " ATRÁS"

        elif left_state == "CLOSED" and right_state == "CLOSED":
            action = " DERECHA"

        elif left_state == "OPEN" and right_state == "OPEN":
            action = " IZQUIERDA"

        else:
            action = "—"

        print(
            f"Izq: {left_state} | Der: {right_state}  =>  {action}"
        )


conn = leap.Connection()
conn.add_listener(TwoHandGestureListener())

with conn.open():
    print("Gestos a 2 manos activos (Ctrl+C para salir)")
    while True:
        time.sleep(0.1)
```

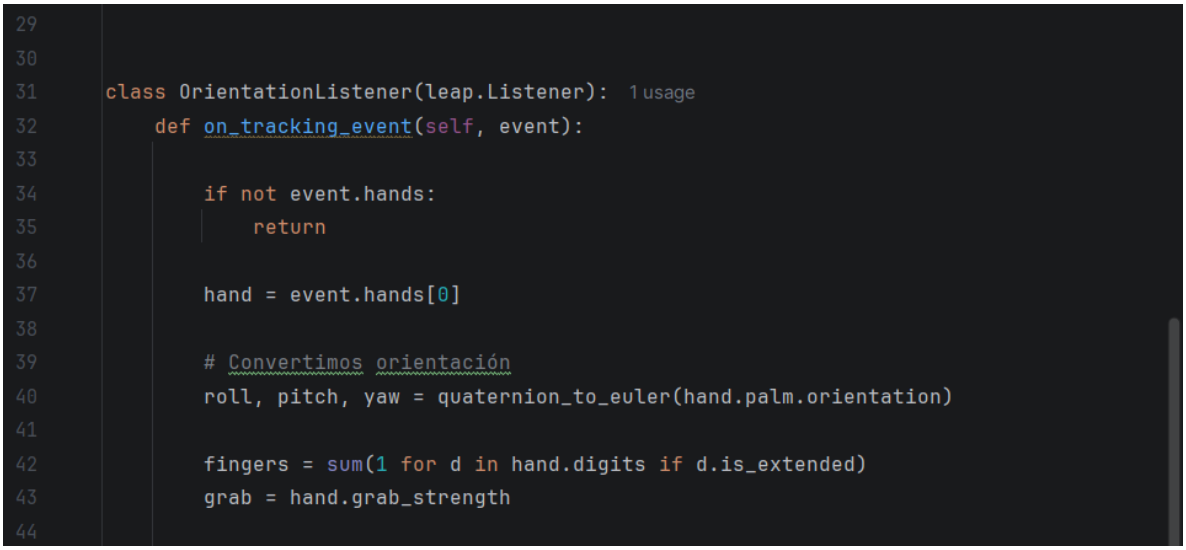
Figura 40
Código test de gestos básicos mano abierta y cerrada.

```python
import leap
import time

class GestureListener(leap.Listener):  1 usage
    def on_tracking_event(self, event):

        if not event.hands:
            print("No hay mano")
            return

        hand = event.hands[0]

        # Contar dedos extendidos
        extended_fingers = 0
        for digit in hand.digits:
            if digit.is_extended:
                extended_fingers += 1

        grab = hand.grab_strength

        # Clasificación simple
        if extended_fingers == 0:
            gesture = "✊ MANO CERRADA"
        elif extended_fingers == 5:
            gesture = "🖐 MANO ABIERTA"
        else:
            gesture = f"{extended_fingers} dedos"

        print(
            f"Gesto: {gesture} | "
            f"Dedos: {extended_fingers} | "
            f"Grab: {grab:.2f}"
        )

conn = leap.Connection()
conn.add_listener(GestureListener())

with conn.open():
    print("Prueba de gestos activa (Ctrl+C para salir)")
    while True:
        time.sleep(0.1)
```

Figura 41
Código test de gestos para movimientos.



```python
import leap
import time
import math

PITCH_THRESHOLD = 0.35
ROLL_THRESHOLD  = 0.35

def quaternion_to_euler(q):  1 usage
    x, y, z, w = q.x, q.y, q.z, q.w

    # Roll (X)
    sinr_cosp = 2 * (w * x + y * z)
    cosr_cosp = 1 - 2 * (x * x + y * y)
    roll = math.atan2(sinr_cosp, cosr_cosp)

    # Pitch (Y)
    sinp = 2 * (w * y - z * x)
    if abs(sinp) >= 1:
        pitch = math.copysign(math.pi / 2, sinp)
    else:
        pitch = math.asin(sinp)

    # Yaw (Z)
    siny_cosp = 2 * (w * z + x * y)
    cosy_cosp = 1 - 2 * (y * y + z * z)
    yaw = math.atan2(siny_cosp, cosy_cosp)

    return roll, pitch, yaw
```



```python


class OrientationListener(leap.Listener):  1 usage
    def on_tracking_event(self, event):

        if not event.hands:
            return

        hand = event.hands[0]

        # Convertimos orientación
        roll, pitch, yaw = quaternion_to_euler(hand.palm.orientation)

        fingers = sum(1 for d in hand.digits if d.is_extended)
        grab = hand.grab_strength

```

```
        if grab > 0.8:
            estado = "✊ STOP"
        elif fingers >= 4:
            estado = "🖐 ACTIVO"
        else:
            estado = "⚠️ INDEFINIDO"

        movimiento = "CENTRO"

        if estado == "🖐 ACTIVO":
            if pitch < -PITCH_THRESHOLD:
                movimiento = "⏩ ADELANTE"
            elif pitch > PITCH_THRESHOLD:
                movimiento = "⏪ ATRÁS"
            elif roll > ROLL_THRESHOLD:
                movimiento = "➡️ DERECHA"
            elif roll < -ROLL_THRESHOLD:
                movimiento = "⬅️ IZQUIERDA"

        print(
            f"{estado} | {movimiento} | "
            f"Pitch={math.degrees(pitch):.1f}° "
            f"Roll={math.degrees(roll):.1f}°"
        )

conn = leap.Connection()
```

```
conn.add_listener(OrientationListener())

with conn.open():
    print("Gestos por orientación activos (Ctrl+C para salir)")
    while True:
        time.sleep(0.1)
```

Figura 42
Código fuente del módulo de pruebas para la adquisición de datos del sensor Leap Motion.

```python
import leap
import time

class HandListener(leap.Listener):  1 usage
    def on_tracking_event(self, event):
        if not event.hands:
            print("No hay mano")
            return

        hand = event.hands[0]

        palm = hand.palm.position
        grab = hand.grab_strength

        print(
            f"Mano detectada | "
            f"Grab: {grab:.2f} | "
            f"Palm x={palm.x:.1f}, y={palm.y:.1f}, z={palm.z:.1f}"
        )

conn = leap.Connection()
conn.add_listener(HandListener())

with conn.open():
    print("Mueve la mano (Ctrl+C para salir)")
    while True:
        time.sleep(0.1)
```

Figura 43
Diseño del robot humanoide y robot móvil.

Figura 44
Diseño del humanoide bípedo.

Figura 45
Diseño del robot móvil.

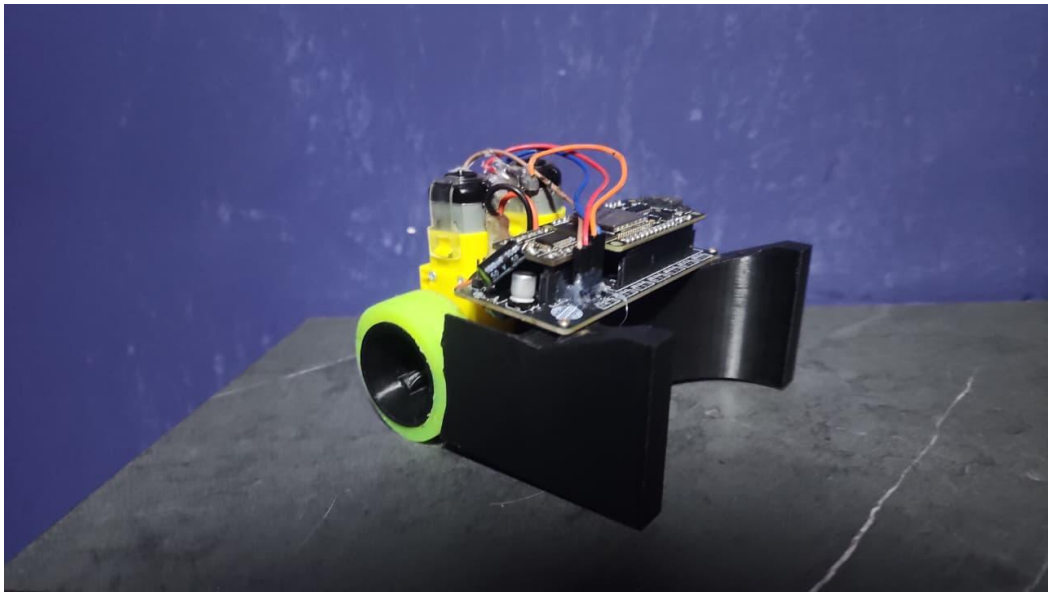


Figura 46
Humanoide y robot móvil.